VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

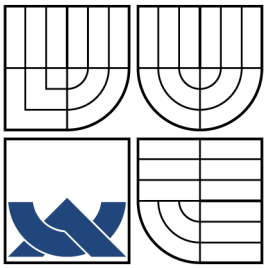
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

ZVUKOVÝ MODUL PRO VÝVOJOVÝ KIT

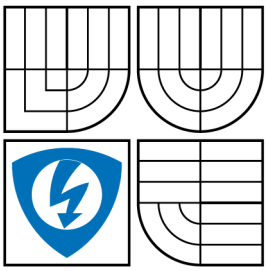
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JIŘÍ PRIŠKIN
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

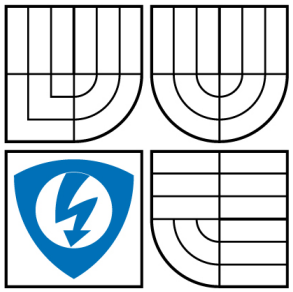
ZVUKOVÝ MODUL PRO VÝVOJOVÝ KIT
SOUND MODULE FOR EVALUATION KIT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ PRIŠKIN
AUTHOR

VEDOUCÍ PRÁCE — ING. PETR SYSEL, PH.D.
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Priškin Jiří	***ID:***	74590
Ročník:	3	***Akademický rok:***	2007/2008

NÁZEV TÉMATU:

Zvukový modul pro vývojový kit

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce je navrhnout modul pro zpracování zvukových signálů pro vývojový kit ATSTK1000 firmy Atmel a zprovoznit jej s ovladači pro dodávaný operační systém Linux. Prostudujte vývojový kit ATSTK1000 a jeho rozhraní pro připojení vnějších obvodů. Navrhněte obvodové zapojení zvukové karty, která bude s procesorem komunikovat pomocí rozhraní AC97. Navrženou kartu realizujte a zprovozněte ji s ovladači ALSA v dodávaném operačním systému Linux. Funkčnost ověřte testovací aplikací (např. softwarový IP telefon), který bude navrženou zvukovou kartu využívat.

DOPORUČENÁ LITERATURA:

[1] AVR32 32-bit Microcontroler [online]. Atmel. 2007. [cite 11.10.2007]. Dostupné na URL http://www.atmel.com/dyn/resources/prod_documents/doc32003.pdf
[2] AT32STK1000 Schematic [online]. Atmel, 2006. [cite 11.10.2007]. Dostupné na URL http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3918
[3] Audio Codec '97 Revison 2.1 [online] Intel Corporation, 1998. [cite 11.10.2007]. Dostupné na URL http://freenet-homepage.de/kxdev/docs/codecs/ac97r21.pdf

Termín zadání:	11.2.2008	***Termín odevzdání:***	4.6.2008

Vedoucí práce: Ing. Petr Sysel, Ph.D.

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

UPOZORNĚNÍ:

PRVNÍ LIST LICENČNÍ SMOUVY

DRUHÝ LIST LICENČNÍ SMOUVY

ABSTRAKT

Tato práce se zabývá návrhem a zprovozněním zvukového modulu pro vývojový kit Atmel ATSTK1000. Úvodem je seznámení se s vlastnostmi, technickým a programovým vybavením vývojového kitu sloužícího k vývoji embedded zařízení založených na AT32AP7000. Mikrořadič AP7000 je jednočipový systém s mikroprocesorovým jádrem AVR32, obsahující další podpůrné obvody, řadiče a rozhraní kterými samotné jádro nedisponuje. Ve vývojovém kitu ovšem není žádný analogově-číslicový převodník pro zpracování zvukového signálu. To vedlo k realizaci přídavného modulu plnícího tuto úlohu. Část práce popisuje standard Intel AC'97 včetně rozhraní AC-link, které je využito k plně duplexnímu přenosu dat mezi realizovaným zvukovým modulem a vývojovým kitem STK1000. Zvukový modul je řešen v podobě zásuvné karty. Poslední kapitolou je průběh instalace operačního systému Linux jako spustitelného obrazu paměťové karty Flash SD/MMC a testování zvukového modulu pod architekturou ALSA.

KLÍČOVÁ SLOVA

Zvukový modul, STK1000, AP7000, AVR32, Embedded Linux, Buildroot, ALSA, AC'97, AC-link, Kodek, Řadič, AD1886A

ABSTRACT

This thesis is dealing with design and putting into operation of sound module for evaluation kit Atmel ATSTK1000. At the beginning, there is an introduction with properties, hardware and software equipment of the evaluation kit which is used for development of embedded devices based on AT32AP7000. The microcontroller AP7000 is a complete system on chip with an AVR32 microprocessor core, containing additional supporting circuits, controllers and interfaces which the core itself does not have included. However, there is no digital to analog converter for audio signal processing. This has led to design an additional module performing this task. A part of thesis describes Intel AC'97 standard including AC-link interface, which is used for full-duplex data transmission between realized sound module and the evaluation kit STK1000. The sound module is designed as a plug-in card. The last chapter presents the Linux operating system installation process as an SD/MMC Flash memory card boot image creation and test of the sound module under ALSA architecture.

KEYWORDS

Sound Module, STK1000, AP7000, AVR32, Embedded Linux, Buildroot, ALSA, AC'97, AC-link, Codec, Controller, AD1886A

PRIŠKIN, J. *Zvukový modul pro vývojový kit.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 58 s. Vedoucí bakalářské práce Ing. Petr Sysel, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Zvukový modul pro vývojový kit“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Na tomto místě bych rád poděkoval vedoucímu bakalářské práce Ing. Petru Syslovi, Ph.D. za technickou pomoc, vstřícnost a věcné připomínky při vypracování této bakalářské práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Cílem této práce je návrh a realizace přídavné zvukové karty pro vývojový kit Atmel ATSTK1000 a její zprovoznění s aplikacemi pod operačním systémem Linux.

K dosažení tohoto cíle je nejdříve nutné prozkoumat technické vybavení vývojového kitu, převážně z hlediska možnosti připojení vnějších periferních zařízení a rozhraní použitelná pro přenos toku zvukových dat. Až bude vývojový kit poznán z tohoto pohledu, bude dalším krokem seznámení se s jeho dodávaným programovým vybavením a to zase se zřetelem na možnosti konfigurace ovladačů zvukových karet v Linux BSP (Board Support Package).

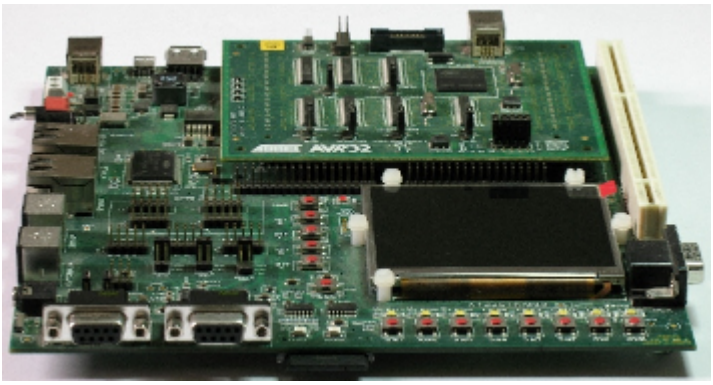
Potom bude následovat rozbor standardu zvukové architektury Audio Codec '97 (AC'97) dle specifikace Intel®, protože právě rozhraní AC-link specifikace AC'97 bylo zvoleno ke komunikaci mezi STK1000 a zvukovým modulem. Teď už bude možné přistoupit k výběru kodeku, návrhu zapojení, plošného spoje a realizaci zvukového modulu, řešeného jako zásuvná karta do rozširujícího slotu na základní desce STK1000.

Následuje vytvoření spustitelného obrazu souborového systému ext2 (The Second Extended File System), který se nahraje do paměťové karty SD/MMC. Vývojový kit je potom schopen zavést systém z karty SD/MMC pomocí zavaděče U-Boot, uloženém v paměti Flash na STK1000. Ve vytvořeném obrazu musí jádro Linux obsahovat ovladač zvukové karty standardu AC'97, také s podporou použitého kodeku. Zde bude využíván zvukový podsystém ALSA (Advanced Linux Sound Architecture).

Posledním úkolem je pod tímto systémem zprovoznit zhotovený zvukový modul a otestovat ho s aplikacemi Linux.

1 VÝVOJOVÝ KIT STK1000

Vývojový kit je určen k návrhu a implementaci embedded (vestavěných) systémů využívajících mikrořadič AT32AP7000. Nejen že zpřístupňuje rozhraní, kterými je vybaven AP7000 ke snadno připojitelným konektorům, ale představuje vlastně kompletní referenční systém s nadmnožinou vybavení periferiemi jakými může disponovat prototyp vyvíjeného zařízení.



Obr. 1.1: Vývojový kit STK1000.

K tomuto účelu jsou na základní desce STK1000 součásti a periferie rozšiřující možnosti mikrořadiče AP7000:

- paměti 8MB SDRAM a 8MB Flash,
- slot pro paměťovou kartu SD/MMC (například kartu se spustitelným souborovým systémem operačního systému GNU/Linux dodanou ke kitu),
- slot pro paměťovou kartu Compact Flash (CF),
- zobrazovač LCD o rozlišení QVGA (320x240),
- konektor výstupu VGA na externí monitor,
- dva konektory rozhraní PS/2 pro připojení klávesnice a myši,
- dvojitý vysílač/přijímač fyzické vrstvy sítě Fast Ethernet 10/100 Mbps a dvě zásuvky RJ45,
- konektory rozhraní USB a UART (standard RS-232),
- vysílač/přijímač infračerveného přenosu dle specifikace IrDA,
- tlačítka ke generování signálů RESET a přerušení procesoru,
- tlačítka a diody LED k univerzálnímu uživatelskému použití,
- dvouřadé konektory na plošném spoji (headers) GPIO umožňující přístup k vybraným signálům pro obecné použití,
- propojky s možnostmi nastavení některých periferií,

- vnější stereofonní číslicově-analogový převodník (DAC) se zesilovačem. Pomocí propojky lze volit výstup do zásuvky stereo Jack 3,5 mm mezi vnějším a vnitřním DAC. Popis zvukových možností je v kapitole 1.3 ,
- stabilizátory s filtracemi potřebných napájecích napětí pro celý systém.

Na základní desce je také rozšiřující přímý konektor podobný slotu PCI známého z osobních počítačů IBM PC, avšak není s tímto slotem kompatibilní! Hlavním účelem tohoto konektoru je snadná rozšiřitelnost systému libovolnou přídavnou kartou schopnou pracovat se signály zpřístupněnými konektorem. Na jeho kontakty jsou vedeny signály z dceřinné i základní desky a to včetně sériových i paralelních rozhraní které nemají zvlášť k tomu určený konektor na základní desce. Pro úplnou představu o tom, jaká rozhraní jsou přístupná přes tento konektor je v příloze B uvedeno jeho přesné zapojení.

Hardware vývojového kitu kromě základní desky STK1000 obsahuje dceřinnou desku STK1002 s paticí BGA256 pro samotný mikrořadič AP7000. Obě desky jsou navzájem propojeny dvěma konektory. Dceřinná deska dále obsahuje:

- krystalové rezonátory ke generování taktovacích kmitočtů pro mikrořadič, jeho sběrnice a rozhraní,
- propojky k připojení rozhraní na periferie a k měření spotřeby proudu různých napájených částí mikrořadiče,
- tvarovací obvody signálu RESET a další pomocné obvody mikrořadiče,
- ladící rozhraní JTAG,
- konektor Nexus,
- konektor rozhraní USB.

Další informace k vývojovému kitu STK1000 můžete najít v uživatelské příručce [1] , ze které jsem také vycházel. Konkrétnější představu o tom, co všechno základní deska obsahuje a jak jednotlivé části spolupracují, mě potom dalo její schéma zapojení [2] . Dokumentace k STK1000 je volně dostupná na stránce produktu [3] .

1.1 Mikrořadič AT32AP7000

Mikrořadič AP7000 je aplikačním procesorem, jehož architekturou je kompletní jednočipový systém (System-on-chip), kde jsou vedle samotného procesoru AVR®32AP také další podpůrné obvody, řadiče a rozhraní, umožňující vyvíjet zařízení s co nejmenším počtem součástek. AP7000 ve vnitřní struktuře doplňuje AVR32 AP o tyto další obvody:

- vysokorychlostní vnitřní sběrnice a mosty propojující jednotlivé vnitřní bloky,
- vnější sběrnice rozhraní pamětí SDRAM a Flash,
- rozhraní pro paměťovou kartu SD/MMC,
- generátory taktovacích kmitočtů,
- časovače a čítače,
- řadiče vnějších přerušení,
- řadiče přímého přístupu do paměti (DMA),
- řadič zobrazovače LCD,
- koprocesor obrazových bodů pro rychlé vektorové výpočty,
- rozhraní k zachytávání obrazu do rozlišení 2048x2048,
- rozhraní PS/2,
- dvoudrátové rozhraní (TWI) podporující standard Philips I^2C,
- sériová rozhraní USB, USART a SPI,
- řadič synchronní sériové komunikace (SSC) konfigurovatelný pro různé typy rozhraní, například standard Philips I^2S,
- podvrstva řízení přístupu k médiu (MAC) sítě Ethernet,
- řadič AC '97 specifikace Intel® pro audio kodeky, viz kapitola 2.1 ,
- stereofonní číslicově-analogový převodník,
- ladící rozhraní JTAG,
- řadič pulsní šířkové modulace (PWM).

Mikroprocesorové jádro AVR32 AP 32-bitové architektury RISC je optimalizováno pro využití v aplikacích vyžadujících nízkou spotřebu současně s velkým početním výkonem, například pro zpracování obrazu nebo zvuku, což je právě případ přenosných embedded systémů jako jsou multimediální přehrávače, telefony, GPS a další. Splnění těchto protichůdných požadavků je dosaženo optimalizací operací na co nejmenší počet strojových cyklů. AVR32 je schopen provádět náročné multimediální výpočty při hodinovém kmitočtu pouhých 150 MHz. U procesoru dochází k menšímu úniku tepla a samozřejmě potom k menší spotřebě. Větší průchodnosti procesoru na jeden strojový takt se dosahuje lepší efektivitou mikrokódu zpracování instrukce v CPU, současným zpracováváním více na sobě nezávislých instrukcí mimopořadí, efektivnějším čtením a ukládáním dat. Překladač jazyka C je úzce spjat s architekturou AVR32 a proto maximálně využívá optimalizace rychlosti a velikosti kódu. Díky instrukční sadě s různou délkou operandů některých instrukcí, závisející na adresovacím režimu, je schopen kompilátor generovat kód velké hustoty. Kompaktnější kód než u konvenčních mikroprocesorů zvládá stejnou úlohu vedle příznivého poměru výkon/spotřeba také s menšími paměťovými nároky. Instrukční

sada obsahuje rozšíření o DSP instrukce s podporou aritmetiky se saturací a velkou rozmanitostí instrukcí násobení. Dalším rozšířením je SIMD, využitelné pro multimediální aplikace. V AVR32 AP je navíc implementovaná hardwarová akcelerace interpretu mezikódu jazyka Java, virtuální stroj Java (JVM). Procesor je možno přepínat mezi režimy RISC nebo Java.

V dokumentu [4] je toho mnohem více o mikrořadiči AP7000, ale jeho detailnější popis není cílem této práce. Na stránce produktu výrobce Atmel® [5] je uložena další dokumentace k AP7000.

1.2 Programové vybavení

Jeden z faktorů při rozhodování, kterou platformu použít pro návrh embedded systémů je vedle početního výkonu, spotřeby, typu a počtu rozhraní mikrořadiče také dostupné hardwarové a softwarové vývojové prostředí od výrobce procesoru. S vývojovými deskami většina výrobců také dodává podpůrné balíčky (BSP – Board Support Package) k usnadnění přechodu na novou hardwarovou platformu a poskytnutí potřebných nástrojů pro práci se systémem. Nejdůležitějšími částmi BSP je firmware s nezbytnými rutinami pro inicializaci a nastavení parametrů konkrétní verze procesoru, řadičů a sběrnic, potom je to zavaděč, jehož součástí je velmi často terminálové připojení, realizované sériovým rozhraním RS-232 nebo přístupem ze sítě TCP/IP, umožňující práci se systémem a nahrávání spustitelných obrazů operačních systémů na zaváděcí zařízení, například do paměti Flash, ze které potom při startu samotný zavaděč nahraje konkrétní obraz do operační paměti a jej spustí. Kromě výše zmíněných základních komponent BSP jsou pro vývoj dodávány další součásti jako jsou překladače programovacích jazyků a ladící nástroje, manuály, katalogové listy součástek a ostatní dokumentace pro vývojáře. Hodně výrobců také zahrnuje do softwarového balíčku celou distribuci operačního systému (OS) GNU/Linux s už upraveným jádrem, moduly a knihovnami pro konkrétní hardware.

Firma Atmel není v této věci pozadu a k deskám vývojového kitu STK1000 dodává CD s podpůrným softwarem a paměťovou kartu Flash SD 256 MB obsahující obraz s funkčním souborovým systémem ext2 operačního systému Linux BSP. Nejnovější verze obrazů CD „AVR32 Linux BSP“[1] a paměťové karty SD/MMC s OS Linux BSP jsou na domovské stránce vývojového kitu [3] .

1.2.1 Obraz firmware a zavaděče

Vývojový kit používá firmware/zavaděč U-Boot od DENX Software Engineering, který je poměrně často používán v embedded zařízeních. Dle potřeby lze tímto ob-

[1] Podpůrný balíček „AVR32 Linux BSP“ byl dříve znám pod názvem „STK1000 BSP“.

razem nahradit stávající, který je uložen v paměti Flash na desce STK1000.

1.2.2 Obraz OS Linux BSP

Součástí BSP je obraz paměťové karty Flash SD s kompletním souborovým systémem ext2 předinstalovaného OS AVR32 Linux BSP. Aktuální verze OS Linux BSP je postavena na jádře 2.6.18. Mnoho distribucí Linux pro embedded zařízení včetně této využívá zvláštní spustitelný soubor BusyBox poskytující minimalistickou náhradu za mnoho běžných nástrojů UNIX/Linux, například: `cat`, `chgrp`, `chmod`, `chown`, `cp`, `dd`, `df`, `dmesg`, `du`, `find`, `grep`, `gunzip`, `gzip`, `hostname`, `id`, `init`, `insmod`, `kill`, `killall`, `ln`, `ls`, `lsmod`, `mkdir`, `mkfifo`, `more`, `mount`, `mv`, `nslookup`, `ping`, `ps`, `pwd`, `reboot`, `rm`, `rmdir`, `rmmod`, `sed`, `sh`, `syslogd`, `tail`, `tar`, `telnet`, `touch`, `umount`, `uname`, `whoami`, `zcat` a podobně. Místo plnohodnotných spustitelných souborů jsou v adresářích `bin` a `sbin` většinou jen symbolické odkazy na spustitelný soubor `busybox` a jejich názvy jsou totožné se jmény původních souborů. Samotný BusyBox bývá často pro embedded distribuce sestaven s menší standardní knihovnou uClibc namísto s GNU Libc. Nahrazované funkce původních programů kolekcí BusyBox jsou sice poněkud ochuzenější o některé možnosti ale to není v embedded systémech na závadu. Naopak je zde zřetelný přínos BusyBoxu v úspoře místa na disku a systémových prostředcích.

1.2.3 Vývojové a ladící nástroje

Základem je sada nástrojů toolchain podporující architekturu AVR32. Toolchain obsahuje sbírku křížových (cross) překladačů GCC (zejména jazyka C/C++), sadu binutils s dalšími nástroji (např. assembler nebo linker) k vytváření knihoven nebo spustitelných programů a standardní knihovnu uClibc jazyka C, se kterou se výstupní binární soubory pro cílové embedded zařízení sestavují. GCC potom řídí celý proces přeměny zdrojového kódu na binární spustitelné soubory nebo knihovny. Dle parametrů, kterými je `avr32-gcc` volán, může například jen přeložit z jazyka C do jazyka symbolických adres (Assembler). Ale také může potom vyvolat překladač `avr32-as` k přeložení kódu Assembler do binárního objektového souboru a následně za pomocí sestavovacího programu (linker) `avr32-ld` ze všech objektových souborů vytvořit spustitelný binární soubor. Jak už bylo zmíněno překladač `avr32-as` a linker `avr32-ld` jsou součástí sbírky binárních nástrojů binutils. Neocenitelným pomocníkem při vývoji softwaru je ladící program (debugger). V balíčku je k dispozici GNU debugger GDB, který dohromady s dodaným serverem `avr32gdbproxy` dovoluje testovat a lokalizovat chyby v cílovém embedded systému vzdáleně z připojeného osobního počítače. Nakonec můžeme mezi vývojové nástroje zařadit různé

knihovny funkcí, programy pro přípravu a nahrávání obrazů na cílový systém, rozmanité konvertory formátů, skripty zjednodušující často opakované úkony a další. Nástroje jsou připraveny pro instalaci do OS Linux (distribuce Fedora, Ubuntu Linux, openSUSE), Windows za použití prostředí Cygwin (emulátor OS Linux), nebo je možné ručně instalovat jednotlivé části do dalších platforem.

1.2.4 Zdrojové soubory

Dodávané zdrojové soubory jádra a knihoven OS Linux umožňují přizpůsobovat jej konkrétním požadavkům a přidávat další možnosti systému, kdy původně předinstalovaný OS bývá většinou kompromisem mezi nároky na místo a rychlostí na jedné straně nebo možnostmi vybavení na druhé. Také různé nástroje jsou dodávány se zdrojovými soubory pro případ, že instalace nepodporují požadovanou platformu a vznikne potřeba si sestavit vlastní binární soubory ze zdrojových.

1.2.5 Dokumentace

Tam lze zařadit: uživatelské příručky a návody (stránky Wiki), návody k řešení vybraných programátorských problémů včetně příkladů v podobě zdrojových souborů jazyka C, katalogové listy nejdůležitějších součástek, přehled použitého materiálu základní desky STK1000 a dceřinné STK1002.

1.2.6 Soubory pro desku NGW

Testovací deska NGW (Network Gateway) je levnější variantou vývojového kitu s mikrořadičem AP7000. Je vybavena menším množstvím periferií než STK1000 a je zaměřena více na síťové aplikace. V balíčku jsou obrazy firmware/zavaděče U-Boot určeny právě pro tuto desku stejně jako některé stránky Wiki BSP. Mnoho jiných součástí v BSP je společných s STK1000.

1.3 Zvukové možnosti

Vývojový kit nedisponuje žádným analogově-číslicovým převodníkem (ADC), nemá tedy zvukový vstup. Stereofonní výstup je do zásuvky Jack 3,5 mm volitelný pomocí propojky mezi vnitřním DAC mikrořadiče AP7000 nebo vnějším na základní desce STK1000. Na desce STK1000 je také reproduktor připojený na výstup výkonového monofonního zesilovače vnějšího DAC.

1.3.1 Vnitřní číslicově-analogový převodník

Samotný mikrořadič AP7000 má ve své struktuře stereofonní 16-bitový DAC se vzorkovacím kmitočtem do 50 kHz. Paralelní vstupní data jsou ve formátu, kde záporné číslo je vyjářeno dvojkovým doplňkem. Výstup z převodníku představuje bitový tok vedený komplementární dvojicí vodičů (kladný a záporný) pro každý kanál. Pomocí jednoduchého filtru typu dolní propust se tento tok dá převést na analogový signál a zbavit šumu nad propustným pásmem. Výsledný audio signál je možno přivést na vstup nízkofrekvenčního zesilovače o vysoké impedanci. Principielně se jedná o dva jednobitové (bitstream) převodníky sigma-delta (Σ-Δ) 3. řádu s koeficientem převzorkování 128. Ještě než se data přivedou k modulátoru Σ-Δ je realizováno nadvzorkování interpolačním filtrem 4. řádu. Za účelem kompenzace kmitočtové charakteristiky interpolačního filtru je před něj zařazen vyrovnávací filtr FIR. Celková kmitočtová charakteristika převodníku bude potom plochá po celé šířce propustného pásma.

1.3.2 Vnější číslicově-analogový převodník

Na základní desce STK1000 je další převodník realizován integrovaným obvodem AT73C213. Tento obvod je původně určen pro přenosná zařízení napájená z baterií vyžadující zvukový výstup (telefony, přehrávače, kapesní počítače, bezdrátová sluchátka ...). DAC je 20-bitový, s nejvyšším vzorkovacím kmitočtem 48 kHz. Vstupní zvuková data jsou přiváděná sériovým rozhraním I^2S, které je na základní desce STK1000 zprostředkováno signály SSC_A mikrořadiče AP7000. Šířka zvukového formátu (16, 18 a 20 bitů), hlasitost, funkce mute a další jsou programovatelné po rozhraní SPI. Výstup je zesílen ještě v AT73C213 stereofonním nízkofrekvenčním zesilovačem 20 mW o zatěžovací impedanci 32 Ω a na desce STK1000 je spojen s propojkou pro výběr mezi vnějším a vnitřním převodníkem do zásuvky Jack. V AT73C213 je navíc monofonní výkonový zesilovač 440 mW o zatěžovací impedanci 8 Ω. Deska STK1000 propojuje vstup tohoto zesilovače s monofonním výstupem převodníku a výstup výkonového zesilovače je potom přiveden na reproduktor umístěný na základní desce.

1.3.3 Podpora zvuku v BSP

V operačním systému Linux BSP jsou dva standardní podsystémy pro přístup programového vybavení ke zvukovému hardware přes aplikační rozhraní (API – Application Programming Interface). Starším z nich je OSS (Open Sound System). Pro tento zvukový podsystém je v BSP k dispozici ovladač vnitřního DAC `at32dac`. Modernější ALSA poskytuje flexibilnější zvukovou architekturu operačního systému

Linux. Má zdokonalené API a také větší podporu zvukových karet. Její součástí je také emulátor OSS pro zpětnou kompatibilitu se staršími multimediálními aplikacemi. Ovladač vnějšího DAC `snd-at73c213` je už napsán pro zvukový podsystém ALSA.

2 ZVUKOVÝ MODUL

Z popisu zvukových možností vyplývá, že vývojový kit STK1000 disponuje hned dvěma způsoby, jak převést data na analogový audio signál, systém obsahuje jeden vnější a jeden vnitřní číslicově-analogový převodník (DAC). Proto se dá STK1000 využít také pro aplikace vyžadující zvukový výstup. Můžeme se ale setkat také s potřebou, mít možnost přivést někam na vstup analogový signál, například pokud bychom chtěli ve vývojovém kitu implementovat služby přenosu hlasu (VoIP). Od telefonie se očekává obousměrná komunikace a proto každá koncová stanice, kromě dekódování příchozího datového toku na analogový audio signál, musí umět také opačně kódovat analogový signál z mikrofonu na odchozí datový tok. K tomu je na začátku řetězce nezbytný analogově-číslicový převodník (ADC) k digitalizaci analogového signálu. Nabízí se proto možnost zhotovit jednoduchý rozšiřující modul s ADC, který bude provádět digitalizaci a data z převodníku se pošlou přes vybrané rozhraní na zpracování do systému v STK1000. Aplikace, která bude mít na starosti zpracování zvukových dat z externího modulu, musí mít ovšem softwarovou podporu pro toto zařízení a to včetně obsluhy rozhraní pro komunikaci s převodníkem. To by vyžadovalo další úsilí napsat speciální ovladač jen pro tento proprietární modul ADC, aby byl schopen spolupráce s existující zvukovou architekturou operačního systému. Je však zřejmé, že mnohem výhodnější bude využít nějakého podporovaného řešení.

Někteří výrobci polovodičových součástek nabízí integrované obvody, které v sobě sdružují funkci kódování a dekódování analogového audio signálu většinou ve stereofonní kvalitě. Tyto tzv. kodéry-dekodéry (kodeky) obsahují ve své struktuře vedle ADC a DAC, také linkové a mikrofonní předzesilovače, výstupní zesilovače (oddělovací, výkonové), analogové obvody pro směšování signálů z více vstupů a řízení zisku, číslicové zpracování signálu, taktovací generátor nezbytný k činnosti kodeku a datové komunikaci. Většina masově vyráběných kodeků komunikuje s hostitelským systémem přes standardizovaná datová rozhraní. Výhodou tohoto řešení je kromě hardwarové kompatibility vysoká škála programového vybavení, počínaje ovladači a konče celými multimediálními aplikacemi.

Porovnáme-li možnosti vývojového kitu STK1000 s typy datových rozhraní běžně vyráběných kodeků máme na výběr tyto varianty:

- sběrnice I^2S pro přenos zvukových dat společně s řídícím rozhraním I^2C nebo SPI,
- rozhraní AC-link architektury AC '97,
- rozhraní USB.

Přes výše jmenovaná rozhraní lze připojit kodek přímo k STK1000 aniž by byl potřeba přídavný řadič. Teď už jen zbývá se podívat jaká je softwarová podpora pro jednotlivé varianty připojení. V tomto případě se zde nabízí použít ze světa osobních počítačů rozšířenější kodek AC '97 a dokonce je také v dodaném OS Linux BSP zahrnut ovladač řadiče AC '97 `snd-atmel-ac97` pro zvukový podsystém ALSA.

2.1 Specifikace AC '97

V následujících odstavcích se budu opírat hlavně o originální dokument [6] napsaný původně vývojovou skupinou Intel Architecture Labs (IAL) v roce 1997. Společnost Intel® dnes již místo starší specifikace AC '97 podporuje modernější Intel® High Definition Audio (Intel® HD Audio).

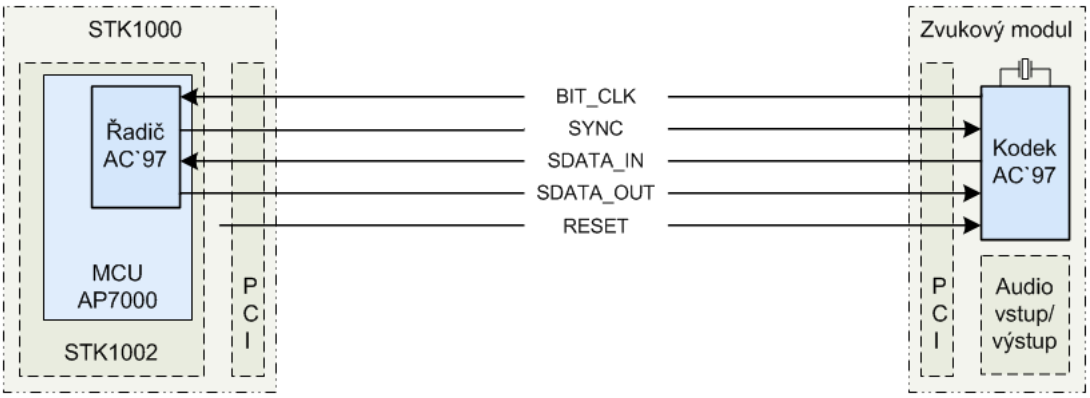
Specifikace AC '97 popisuje integraci zpracování zvuku a modemu do osobních počítačů. Její součástí jsou požadavky na jednotlivé komponenty architektury a způsob komunikace mezi nimi pomocí číslicového rozhraní AC-link. Typická zařízení AC '97 mohou být: audio kodek (AC '97), modemový kodek (MC '97) nebo kombinovaný (AMC '97). Nás samozřejmě zajímá hlavně audio kodek, k němuž jsou ve specifikaci přesně definovány povinné a volitelné vlastnosti, které musí kodek splňovat. Minimální požadavky jsou 16-bitový plně duplexní stereofonní zvuk se vzorkovacím kmitočtem 48 kHz, až 4 stereofonní a 2 monofonní linkové vstupy, analogový vstup z CD, mikrofonní vstup se zesílením 20 dB, programovatelnou regulací zisku a potlačením ozvěny, samostatný linkový stereofonní výstup, konfigurovatelný pomocný stereofonní výstup, monofonní výstup pro hlasitý telefon, úplnou podporu správy napájení, průmyslový standard 48-vývodového pouzdra QFP integrovaného obvodu kodeku s doporučeným zapojením vývodů. Dále jsou v dokumentu vyjmenovány volitelné vlastnosti a podobně ještě pro modemový kodek. V jiné části specifikace je definováno chování a možnosti ovladačů v operačním systému určených ke zpřístupnění podpory zvuku a modemu kompatibilní AC '97.

2.1.1 Architektura AC '97

Základní součásti AC '97 jsou kodek, řadič číslicového rozhraní AC-link a rozhraní AC-link připojující jeden nebo více kodeků k řadiči. Dle této specifikace je možné připojit k jednomu řadiči až 4 kodeky. Potom jeden z kodeků je primární. Jeho hodinový kmitočet je využíván řadičem a ostatními (sekundárními) kodeky k bitové synchronizaci rozhraní AC-link pomocí signálu BIT_CLK. V případě navrhovaného zvukového modulu uvažujeme pouze jeden kodek, který musí samozřejmě pracovat jako primární. Také řadič AC '97 je už součástí mikrořadiče AP7000 jak již bylo uvedeno v části 1.1 . Signálové vodiče AC-link jsou vyvedeny z řadiče AC '97 do

rozširujícího slotu na základní desce STK1000, viz jeho zapojení B . Je nutné tedy navrhnout zvukový modul jako zásuvnou kartu obsahující kodek s rozhraním AC-link specifikace AC ‘97.

Následující obrázek 2.1 ukazuje souhrnně architekturu AC ‘97 integrovanou do vývojového kitu STK1000.



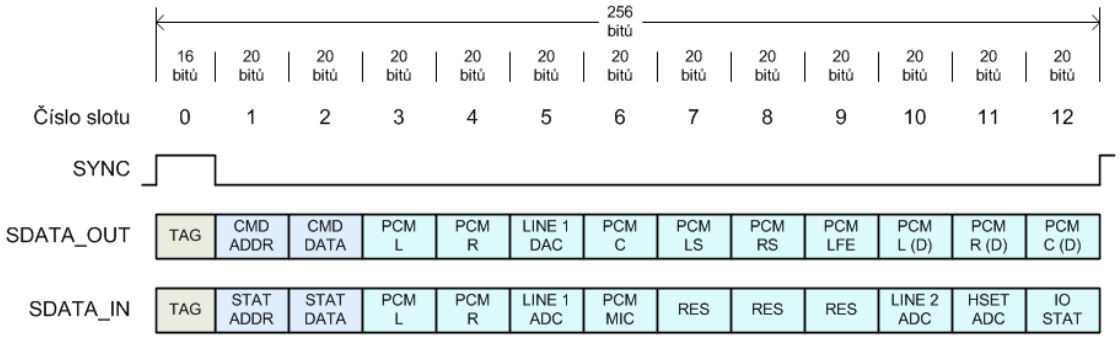
Obr. 2.1: AC ‘97 ve vývojovém kitu STK1000.

Kodek AC ‘97 a řadič mezi sebou komunikují obousměrně přes 5-vodičové rozhraní AC-link. Směr přenosu je definován z pohledu hostitelského systému, to znamená že, vstupní datový tok je směrem od kodeku k řadiči a výstupní naopak.

2.1.2 Rozhraní AC-link

Fyzicky je synchronní sériové rozhraní AC-link tvořeno dvěma vodiči datového toku (SDATA_IN, SDATA_OUT), bitovou synchronizací (BIT_CLK), rámcovou synchronizací (SYNC) a signálem počáteční inicializace (RESET). Datové toky PCM jednotlivých zvukových kanálů jsou společně s informacemi z řídícího registru číslicového rozhraní vkládány do výsledného rámce v pevně daných časových okamžicích (slotech). Jedná se tedy o klasickou techniku časového multiplexu (TDM). V každém směru je přenos rozdělen na 12 datových toků. Jejich umístění do rámce je znázorněno na obrázku 2.2 a přiřazení slotů potom v tabulce 2.1 .

Bitová synchronizace je zajišťována primárním kodekem, který obsahuje krystalový generátor pracující dle doporučení na dvojnásobném kmitočtu než je bitová rychlost. Hodinový kmitočet BIT_CLK je 12,288 MHz, odpovídá bitové rychlosti. Přesnost hodinového kmitočtu primárního kodeku přímo ovlivňuje kolísání zpoždění (jitter) přenosu dat a tím také kvalitu výsledného zvuku. Z bitové synchronizace přicházející od kodeku odvozuje řadič v hostitelském systému rámcovou synchronizaci SYNC. Protože nejvyšší vzorkovací kmitočet přenášených zvukových dat je 48 kHz a v každém rámci se přenese právě jeden kompletní maximálně 20-bitový



Obr. 2.2: Časové sloty rámce rozhraní AC-link podle specifikace [6].

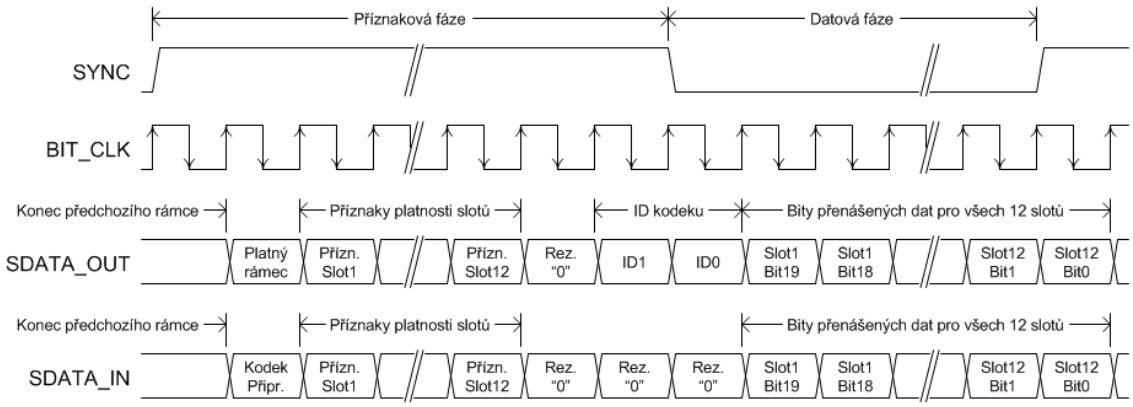
Tab. 2.1: Přiřazení slotů rozhraní AC-link podle specifikace [6].

Výstupní (SDATA_OUT)		
Slot	**Označení**	**Popis**
0	TAG	Příznaky platnosti dat ve slotech a identifikátor kodeku
1	CMD ADDR	Adresa registru a příznak zápisu
2	CMD DATA	16-bitová data k zápisu do registru na danou adresu
3	PCM L	PCM levého předního kanálu (16, 18, 20 bitů)
4	PCM R	PCM pravého předního kanálu (16, 18, 20 bitů)
5	LINE 1 DAC	16-bitová výstupní data modemové linky 1
6	PCM C	PCM středního kanálu (16, 18, 20 bitů)
7	PCM LS	PCM levého surroundového kanálu (16, 18, 20 bitů)
8	PCM RS	PCM pravého surroundového kanálu (16, 18, 20 bitů)
9	PCM LFE	PCM nízkofrekvenčního efektového kanálu (16, 18, 20 bitů)
10	PCM L (D)	Přídavný slot PCM levého kanálu pro dvojnásobnou šířku pásma
11	PCM R (D)	Přídavný slot PCM pravého kanálu pro dvojnásobnou šířku pásma
12	PCM C (D)	Přídavný slot PCM středního kanálu pro dvojnásobnou šířku pásma
Vstupní (SDATA_IN)		
Slot	**Označení**	**Popis**
0	TAG	Příznaky platnosti dat ve slotech
1	STAT ADDR	Adresa čteného registru a příznaky požadovaných slotů
2	STAT DATA	16-bitová přečtená data z registru
3	PCM L	PCM levého kanálu (16, 18, 20 bitů)
4	PCM R	PCM pravého kanálu (16, 18, 20 bitů)
5	LINE 1 ADC	16-bitová vstupní data modemové linky 1
6	PCM MIC	Volitelný třetí vstup ke specializovanému využití pro mikrofon
7–9	RES	Rezervováno k nespecifikovanému využití
10	LINE 2 ADC	16-bitová vstupní data modemové linky 2
11	HSET ADC	16-bitová vstupní data modemové hovorové soupravy
12	IO STAT	Stav modemu

vzorek z každého kanálu, je kmitočet rámcové synchronizace také 48 kHz. Rámce jsou pevné délky, složené vždy dohromady z 13 slotů. Nultý slot je pouze 16-bitový,

dvanáct ostatních pro jednotlivé datové toky jsou délky 20 bitů. Když si spočítáme, že délka jednoho rámce je $16 + 12 . 20 = 256$ bitů a za jednu sekundu se jich přenese $48 . 10^3$, potom hodinový kmitočet bitové synchronizace BIT_CLK skutečně musí být $256 . 48,10^3 = 12,288$ MHz.

Propojení AC-link tedy představuje dvě datové roury, kterými se pro každý směr přenáší konstatně 12 datových 20-bitových slotů rychlostí 48 kHz. K zabránění nejednoznačnosti přenášených dat, teprve sestupnou hranou BIT_CLK příjemce rámce rozhoduje o logickém stavu příslušejícímu dané pozici. Impuls SYNC přechází ze stavu logické „0“ do stavu „1“ s náběžnou hranou BIT_CLK a na sestupnou hranu teprve začíná nový rámec. Proto až následující aktivní hrana BIT_CLK představuje první pozici dalšího rámce. Rámcová synchronizace SYNC setrvá ve stavu logická „1“ během 16 taktů BIT_CLK. Této části se říká příznaková (TAG) fáze, zbytek rámce, kdy je SYNC ve stavu logické „0“, se označuje jako datová fáze. Logické úrovně jednotlivých bitů v časových slotech jsou také určovány vždy až sestupnou hranou příslušejícího pulsu BIT_CLK. Data se ve slotech přenáší ve formátu od nejvyššího bitu (MSB) po bit nejméně významný (LSB) a nevyužité bity jsou zarovnané zprava nulami. Na obrázku 2.3 je vidět nejen časová souslednost začátku rámce a bitových pozic v závislosti na náběžných a sestupných hranách signálu BIT_CLK, ale také detailnější pohled na jednotlivé bity významných částí rámce (příznaková fáze, začátek a konec datové fáze).



Obr. 2.3: Obousměrná komunikace přes rozhraní AC-link, viz dokument [6].

Každý kodek kompatibilní specifikaci AC‘97 obsahuje v sobě 64 16-bitových řídících registrů, ve kterých jsou na pevně stanovených adresách uloženy parametry jednotlivých zvukových vstupů/výstupů a režimy kodeku. Přistupovat lze přitom pouze na sudé adresy a některé z nich jsou jen ke čtení. Ve slotu 1 výstupního rámce SDATA_OUT se dle potřeby vybírá adresa (index) registru (bity 18–12) do kterého se má uložit nová hodnota přenášená ve 2. slotu, nebo je třeba naopak přečíst aktuální stav z registru. O tom, zda se jedná o operaci čtení nebo zápisu do registru kodeku,

rozhoduje první bit (bit 19) ve slotu 1 rámce SDATA_OUT. Jestliže je nastaven na „1“, jde o čtení stavu z registru a kodek musí ve slotu 2 příštího rámce SDATA_IN poslat obsah tohoto registru, současně s indexem tohoto registru ve slotu 1.

Jedním z charakteristických rysů AC ‘97 je možnost nastavit různé délky zvukových vzorků a vzorkovací kmitočet. V případě potřeby přenosu kratších vzorků než 20 bitů (18, nebo 16) se jednoduše nevyužité bity vyplní nulami. Tento mechanismus je obecně používán pro všechny sloty, ve kterých některé bity nenesou řádnou informaci. Větším problémem je podpora různých vzorkovacích kmitočtů, režim VRA (Variable Rate Audio). K tomu je využit také nultý příznakový slot, který pomocí bitů 14–3 indikuje, že byla přiřazena platná data. Každý jednotlivý datový slot má vyhrazen svůj vlastní bit.

Při nižším vzorkovacím kmitočtu se jednoduše v některých rámcích vzorky vynechají a v nultém slotu těchto rámců nebudou nastaveny příslušné příznakové bity. Rozhodující slovo které vzorky vynechávat má vždy kodek. Ten má v sobě vyrovnávací paměť umožňující v pravidelných intervalech (dle aktuálního vzorkovacího kmitočtu) dodávat vzorky do DAC i když v některých rámcích budou vynechány. Když kodek zjistí, že může přijmout další vzorky pro zvukový výstup, požadavek na daný slot signalizuje nastavením příslušného bitu SLOTREQ na hodnotu „0“ v 1. slotu vstupního rámce SDATA_IN. Řadič v příštím výstupním rámci SDATA_OUT, který následuje, naplní požadovaný slot vzorkem PCM. Při konstantní vzorkovací rychlosti 48 kHz jsou bity SLOTREQ trvale na hodnotě „0“ a vzorky PCM jsou odesílány řadičem v každém výstupním rámci. V opačném, vstupním směru je opět na kodeku ve kterých rámcích budou sloty obsahovat platné vzorky a kdy budou prázdné. Platnost dat signalizuje kodek řadiči příslušnými příznakovými bity v nultém slotu rámce SDATA_IN.

V nultém (příznakovém) slotu vstupního rámce SDATA_IN je také hned na první bitové pozici (bit 15), odpovídající první pozici celého rámce, příznak připravenosti kodeku k normální činnosti (Codec Ready). Tento bit je nastaven na „1“ v momentě, kdy je kodek připraven zpracovávat sloty se vzorky PCM. Ve výstupním rámci SDATA_OUT je místo toho tento první bit využit k indikaci validity celého rámce.

2.2 Realizace modulu

Jako základ byl použit integrovaný obvod AD1886A. Tento obvod, od výrobce Analog Devices, Inc. (ADI), sdružuje ve svém 48-vývodovém pouzdře LQFP celou funkcionalitu kodeku, tak jak je předepsáno specifikací AC ‘97. Vyžadovalo to k němu připojit několik vnějších součástek, vstupy/výstupy a získal jsem kompletní zvu-

kový modul, připojitelný přes rozhraní AC-link k vývojovému kitu STK1000. Kodek AD1886A disponuje těmito vlastnostmi:

- 16-bitový stereofonní plně duplexní kodek,
- architektura ADC a DAC s vícebitovými převodníky Σ-Δ,
- dynamický rozsah větší než 90 dB,
- měnitelný vzorkovací kmitočet (režim VRA) od 7040 Hz do 48 kHz, s přesností kroku 1 Hz,
- čtyři stereofonní a dva monofonní vstupy,
- monofonní vstup z mikrofonu s možností zařadit vnitřní předzesilovač se ziskem 10 (20 dB),
- dva stereofonní a jeden monofonní výstup,
- 20-bitový výstup S/PDIF o symbolové rychlosti 32 kHz, 44,1 kHz, nebo 48 kHz,
- oddělené napájení pro číslicové obvody 3,3 V a analogové 5 V,
- oddělený číslicový a analogový společný vodič (zem),
- a dalšími, převážně odpovídajícími požadavkům specifikace AC '97.

Obvodové schéma navrženého zvukového modulu je uvedeno v příloze C. Vycházel jsem z doporučeného zapojení v katalogovém listu [7] integrovaného obvodu AD1886A.

Bitová synchronizace BIT_CLK je odvozena z kmitočtu vnířního oscilátoru, který je řízen krystalem 24,576 MHz. Datové vodiče AC-link včetně signálu RESET s aktivní logickou „0“ jsou přímo napojeny na kontakty do rozširujícího slotu. Výjimkou je bitová synchronizace BIT_CLK, kde jsou hrany hodinového signálu 12,288 MHz na výstupu kodeku omezeny RC členem R1, C14. Přívody napájení jsou blokovány kapacitory 100 nF. Ostatní kapacitory okolo integrovaného obvodu jsou využívány vnitřními filtry kodeku.

Úroveň stereofonního vstupního signálu LINE_IN je snížená napěťovými děliči R2, R3 a R4, R5 na polovinu (-6 dB), protože běžné zdroje audio signálu mají na výstupu nominální efektivní hodnotu napětí 2 V přitom kodek pracuje s úrovněmi 1 V. Naopak signál mikrofonního monofonního vstupu MIC_IN je zesílen operačním zesilovačem IC2 v klasickém invertujícím zapojení. Jeho napěťové zesílení je 10 (20 dB). V případě velmi nízké úrovně signálu mikrofonu je možné pomocí propojky JP1 vybrat na vstup kodeku výstup z IC2 a společně s vnitřním předzesilovačem v AD1886A je zisk celé cesty z mikrofonu do ADC zvětšen na 100 (40 dB). K zajištění stejnosměrného polovičního napětí na neinvertujícím vstupu operačního zesilovače IC2 stejně jako k předpětí na mikrofon se využívá vnitřního zdroje referenčního napětí AD1886A.

Stereofonní výstupy HP_OUT a LINE_OUT jsou téměř rovnocenné s výjimkou toho, že HP_OUT je přizpůsoben pro výstup do sluchátek, čemuž odpovídá minimální zatěžovací impedance 32 Ω, narozdíl od LINE_OUT, na který lze připojit zátěž o minimální impedanci 10 kΩ. Na jeden z rozpínacích kontaktů zásuvky Jack HP_OUT je připojen obvod automatického vypnutí ostatních výstupů pokud je v této zásuvce zasunut konektor, funkce Jack Sense. Do cesty všech audio vstupů a výstupů jsou k oddělení stejnosměrné složky zařazeny vazební kapacitory. Rezistory připojené mezi záporný vývod těchto kapacitorů a zem zajistí správnou polarizaci, vyžadovanou případnými elektrolytickými kapacitory. Kde to bylo možné, snažil jsem se elektrolytickým kapacitorům vyhnout a raději jsem místo nich použil keramické.

Z důvodu elekromagnetické kompatibility se v katalogovém zapojení kodeku doporučuje k blízkosti zásuvek Jack audio vstupů a výstupů přidat filtrační prvky tvořené LC členem. Na těchto místech jsem použil tlumivky v provedení SMD řady BLM 21 od výrobce Murata, speciálně určené k tomuto účelu. Jejich hodnota impedance 600 Ω se udává pro kmitočet 100 MHz. Společné vodiče jsou zase odděleny tlumivkami L9 a L10 řady BLM 41 o impedanci 75 Ω/100 MHz.

Ke zvýšení univerzálnosti zvukového modulu je také vyvedeno rozhraní číslicového audio výstupu S/PDIF. Pomocí vnějšího vysílače/přijímače fyzické vrstvy rozhraní S/PDIF lze potom zvukový modul propojit přes optický nebo metalický koaxiální kabel s jiným zvukovým zařízením podporujícím toto rozhraní.

Motivy obou stran plošného spoje jsou zobrazeny v přílohách D.1 a D.2 v poměru 1:1. Osazení spoje součástkami a jejich seznam, jsou také součástí dokumentace k realizaci zvukového modulu, v přílohách D.3 a D.4 .

3 INSTALACE A TESTOVÁNÍ FUNKČNOSTI

Po osazení součástek zvukového modulu na plošný spoj bylo logickým vyústěním vytvoření obrazu OS Linux BSP s podporou AC'97 v subsytému ALSA a potom mělo nastat ověřování že všechno funguje podle očekávání.

3.1 Instalace Linux BSP

Původní myšlenkou bylo dodržení postupu pro instalaci potřebných nástrojů BSP a následné vytvoření spustitelného obrazu cíle podle návodů v uživatelské příručce [1] přímo od výrobce Atmel. Pokusil jsem se tedy z CD „AVR32 Linux BSP“ nainstalovat potřebný balíček `stk1000bsp-release-2.0.0-2.fc5.noarch.rpm` do hostitelského počítače s operačním systémem Linux distribuce Fedora Core 5. Zjistil jsem však, že instalace neobsahuje přímo potřebné soubory, ale pouze odkazy odkud se mají stahovat z Internetu během samotné aktualizace po načtení konfigurace uložené v depozitáři balíčkovacího systému, například při použití nástroje YUM. Když jsem se pokusil příkazem `yum groupinstall "STK1000 BSP"` spustit instalaci BSP, balíčkovací systém nedokázal najít potřebné soubory na místech, kam odkazovala konfigurace v depozitáři, protože mezitím v době kdy jsem se o to pokoušel, byl archiv na serveru http://www.atmel.no zrušen. Proto jsem musel improvizovat tím, že jsem zkombinoval instalační soubory určené pro jiné distribuce Linux, ale pokaždé jsem měl, zejména během ruční instalace ze zdrojových souborů, problém s některou chybějící sdílenou knihovnou nebo byla starší verze než vyžadovala instalovaná komponenta. Nakonec jsem po různých zásazích do konfigurací, skriptů i souborů Makefile jednotlivé součásti přece jenom doinstaloval, včetně nástrojů toolchain potřebných ke křížovému překladu a sestavení programů pro architekturu AVR32, ale při pokusu o vytvoření obrazu OS Linux se objevila další komplikace a to že v objektových binárních souborech chyběl symbol `__cxa_atexit` během sestavování některých knihoven. Zřejmě zde docházelo ke konfliktu verzí standardní knihovny uClibc a příslušného křížového překladače GCC pro AVR32. Mohly být také špatně nastaveny parametry pro překlad těchto částí kolekce toolchain při volání skriptu `configure`. S ohledem na množství proměnných, které skript při generování souborů Makefile pro GCC a uClibc používá a rizika velké ztráty času při experimetování s nimi, jsem musel najít jiný způsob s jistějším výsledkem jak získat funkční prostředí pro tvorbu aplikací kompatibilních s rodinou mikroprocesorů AVR32 a sestavování spustitelných obrazů pro cílový embedded systém.

Nakonec jsem při procházení různými Internetovými stránkami zaměřenými pro vývoj systémů s architekturou AVR32 objevil řešení a to byl projekt Buildroot.

3.2 Instalace pomocí Buildroot

Oficiální domovskou stránkou projektu Buildroot je http://buildroot.uclibc.org/. Buildroot je balík souborů Makefile a záplat, sloužících dohromady k plně automatickému vytvoření nástrojů toolchain ke křížové kompilaci a následnému sestavení obrazu souborového systému pro cílový systém. Pomocí tohoto projektu je plně automatizován proces generování programového vybavení k vyvíjenému embedded zařízení, který má obvykle jinou architekturu než běžný hostitelský osobní počítač s mikroprocesorem řady Intel x86, na němž je software vyvíjen než bude nahrán do cílového zařízení. Dokonce pokud by byl také cílový systém postaven na architektuře x86 je tu pořád problém s tím, že na hostitelském počítači je velká standardní knihovna GNU Libc, obsahující mnoho funkcí a symbolů, které sdílené knihovny a programy v embedded zařízeních nepotřebují a navíc zde není systémových prostředků nazbyt. Proto je výhodné zajistit, aby se binární soubory pro cílové zařízení sestavovaly s menší knihovnou uClibc, stejně tak aby i jiné knihovny, které budou použity se nekombinovaly s těmi, co jsou v hostitelském systému. Buildroot zajišťuje používání správných souborů tím, že jsou uloženy odděleně v adresářích pro to určených a při kompilaci nebo generování výsledného souborového systému potom pracuje s předdefinovanými cestami k nim. Následující řádky popíší jednotlivé kroky, jak získat spustitelný obraz cíle pomocí Buildroot.

Na adrese http://www.atmel.no/buildroot/buildroot-src.html byl ke ztažení archiv `buildroot-avr32-v2.1.0.tar.bz2`, se kterým jsem konkrétně pracoval. Předpokládal jsem, že zde budou přidány všechny potřebné doplňky a záplaty pro platformu AVR32, respektive specifické součásti týkající se přímo hardwaru vývojového kitu STK1000. V adresáři kam se buildroot rozbalil, jsem zavolal program Make a předal parametr s cílem STK1000:

```
make atstk1002_defconfig
```

Takto se vytvořil soubor `.config` s předdefinovaným nastavením cílového systému STK1000. Vhodné je zkontrolovat, případně upravit obsah tohoto souboru. Nachází se ve výchozím adresáři Buildroot odkud se předtim volal program Make. V následujících řádcích textu budou všechny cesty a názvy souborů relativně vztaženy k tomuto adresáři. Většina implicitních voleb pro architekturu vývojového kitu se zkopírovala ze souboru `target/device/Atmel/atstk1002_defconfig`. K tomuto účelu lze také využít interaktivního prostředí pro nastavení parametrů podobně jako při překladu jádra Linux, například příkazem:

```
make menuconfig
```

Parametry jsou rozděleny do více skupin: obecná nastavení projektu (verze, cesty a názvy společných souborů), cílová architektura, toolchain, verze a cesta ke konfiguračnímu souboru jádra Linux, cílové souborové systémy, BusyBox, nastavení zavaděče U-Boot, volitelné knihovny, programy a další. Následující vybrané řádky konfiguračního souboru jsou typické pro STK1000:

```
BR2_PROJECT="atstk1002"
BR2_avr32=y
BR2_at32ap7000=y
BR2_ARCH="avr32"
BR2_ENDIAN="BIG"
BR2_UCLIBC_CONFIG="target/device/Atmel/uClibc.config.avr32"
BR2_TARGET_ATMEL=y
BR2_TARGET_AVR32=y
BR2_TARGET_AT32AP7000=y
BR2_TARGET_AVR32_ATSTK1002=y
BR2_BOARD_NAME="atstk1002"
BR2_BOARD_PATH="target/device/Atmel/$(BR2_BOARD_NAME)"
```

Některé volby souvisí s přidáním komponent zvukového podsystému ALSA v uživatelském prostoru:

```
BR2_AUDIO_SUPPORT=y
BR2_PACKAGE_ALSA_LIB=y
BR2_PACKAGE_ALSA_UTILS=y
BR2_PACKAGE_ALSA_UTILS_ALSACONF=y
BR2_PACKAGE_ALSA_UTILS_ALSACTL=y
BR2_PACKAGE_ALSA_UTILS_AMIXER=y
BR2_PACKAGE_ALSA_UTILS_APLAY=y
BR2_PACKAGE_ALSA_UTILS_ARECORD=y
BR2_PACKAGE_AUMIX=y
BR2_PACKAGE_LIBSNDFILE=y
```

Sdílených knihoven a programů architektury ALSA je zde ještě více, stejně jako mnoho různých balíčků jiných aplikací, které se mohou přidat pomocí voleb obvykle začínajících prefixem `BR2_PACKAGE_`.

Podobně je nutné také provést konfiguraci jádra Linux. Po automatickém rozbalení archivu se zdrojovými soubory jádra zvolené verze, se po spuštění Make automaticky aplikují záplaty určené právě pro STK1000 a zkopíruje se implicitní konfigurace jádra ze souboru daného parametrem `BR2_PACKAGE_LINUX_KCONFIG` v konfiguraci

Buildroot `.config`. Konkrétně v mém případě se jednalo o výchozí konfigurační soubor:

```
target/device/Atmel/atstk1002/atstk1002-linux-2.6.22.5.config
```

Proto je třeba zkontrolovat a upravit obsah také tohoto souboru. Zde byly pro STK1000 významné tyto volby:

```
CONFIG_AVR32=y
CONFIG_SUBARCH_AVR32B=y
CONFIG_PLATFORM_AT32AP=y
CONFIG_CPU_AT32AP700X=y
CONFIG_CPU_AT32AP7000=y
CONFIG_BOARD_ATSTK1002=y
CONFIG_BOARD_ATSTK1000=y
CONFIG_LOADER_U_BOOT=y
CONFIG_SERIAL_ATMEL=y
CONFIG_SERIAL_ATMEL_CONSOLE=y
CONFIG_AT32AP700X_WDT=m
CONFIG_LCD_LTV350QV=y
CONFIG_FB_ATMEL=y
CONFIG_MMC_ATMELMCI=y
CONFIG_RTC_DRV_AT32AP700X=m
```

Navíc bylo nezbytné také v jádru povolit kompilaci podsystému ALSA, abych mohl testovat zvukový modul. A to nejen aby obsahovalo standardní moduly ALSA, ale také speciální ovladač `snd-atmel-ac97`, určený přímo pro řadič AC'97 který je součástí jednočipového systému AP7000. K aktivaci zvukové podpory v jádře Linux je vhodné zkontrolovat, případně nastavit tyto řádky:

```
CONFIG_BOARD_ATSTK100X_ENABLE_AC97=y
CONFIG_SOUND=y
CONFIG_SND=y
CONFIG_SND_TIMER=m
CONFIG_SND_PCM=m
CONFIG_SND_OSSEMUL=y
CONFIG_SND_MIXER_OSS=m
CONFIG_SND_PCM_OSS=m
CONFIG_SND_PCM_OSS_PLUGINS=y
CONFIG_SND_AC97_CODEC=m
CONFIG_SND_ATMEL_AC97=m
```

Zde se ukázalo prozíravé, že ovladač `snd-atmel-ac97` byl nastaven jako dodatečně zaváděný modul, protože pokud by byl zakompilován přímo do souboru monolitického jádra Linux, potom by se už při inicializaci pokoušel detekovat zvukový modul a po dlouhou dobu než by došlo k vypršení časového limitu pro komunikaci s kartou, by byl celý systém zablokován. Také by nebyla později další příležitost detekovat zvukovou kartu až bude systém spuštěn. Tento problém bude ještě komentován později v části 3.3 až budu popisovat samotné zprovozňování zvukového modulu potom, co jsem vytvořil souborový systém pro cílové zařízení na paměťové kartě SD/MMC.

Po kontrole obou konfiguračních souborů, pro Buildroot i jádro Linux, lze před samotným generováním všech součástí projektu stáhnout z Internetu do lokálního depozitáře, který je umístěn v adresáři `src/dl`, všechny potřebné archivy se zdrojovými soubory jednotlivých balíčků příkazem:

```
make source
```

Je vysoce pravděpodobné že už nemusí být aktuální odkaz na archiv balíčku a je potom třeba zjistit kde jinde je potřebný soubor dostupný a stáhnout ho ručně do adresáře `src/dl`, nebo upravit v příslušném konfiguračním souboru balíčku aktuální odkaz. Například řekněme pro balíček knihovny `libfoo` to bude zřejmě soubor `package/libfoo/libfoo.mk`[1]. V tomto konfiguračním souboru je možné například upravit tyto proměnné, aby se stáhnul správný archiv:

```
LIBFOO_VERSION:=1.6.66
LIBFOO_SOURCE:=libfoo-$(LIBFOO_VERSION).tar.gz
LIBFOO_SITE:=http://ftp.gnu.org/pub/gnu/libfoo
```

Pokud jsou již všechny potřebné soubory v lokálním depozitáři, potom není potřeba být připojen na síť a také není riziko, že v následujícím kroku nebude díky některému nedostupnému balíčku na Internetu přerušen celý proces, který mohl trvat hodiny. Jedná se právě o fázi, kdy jsme připraveni spustit generování všech komponent. Jestliže jsme přeskočili předchozí krok, soubory „.mk“ jednotlivých balíčků se postarají také o stažení chybějících zdrojových archivů, během generování projektu. Mohl jsem tedy konečně odstartovat několikahodinový automatický proces příkazem:

```
make
```

[1] Každý balíček má ve svém podadresáři svůj vlastní soubor s příponou „.mk“ podobného formátu jako normální Makefile. To znamená že jsou zde definice proměnných a hlavně cíle s pravidly obsahující závislosti a příkazy předepisující jak těchto cílů dosáhnout.

Nejdříve se muselo vytvořit prostředí k sestavování binárních souborů pro cílovou architekturu. Proto se vygenerovala kolekce křížových překladačů GCC a binutils. V dalším kroku se rozbalily zdrojové soubory jádra Linux, aplikovaly se na ně záplaty a mohla se přeložit standardní knihovna uClibc za použití správných hlavičkových souborů jádra pro cílový systém. Potom se zkompiloval ladící program GDB, zavaděč U-Boot a samotné jádro Linux. Dále Make procházel v adresáři `package` podadresáře jednotlivých balíčků a podle pravidel definovaných v souborech „.mk“ sestavoval a instaloval komponenty. Aby se rozlišilo, že se jedná o soubory určené pro cílový systém, nikoliv hostitelský, je zde vyhrazen speciální adresář `build_avr32_nofpu/staging_dir` s podobnou hierarchickou adresářovou strukturou jako běžný souborový systém Linux. Také se zde ukládají hlavičkové soubory a knihovny ke kompilaci a sestavování jiných sdílených knihoven nebo programů, určených pro cílovou architekturu. Nedojde tak k tomu, že by se nechtěně zkombinovaly soubory hostitele a cíle.

Nakonec se z kostry souborového systému s požadovanou adresářovou stukturou a základními konfiguračními soubory v `/etc`, jádra Linuxu a jednotlivých aplikací v uživatelském prostoru vytvořily obrazy souborových systémů, které jsou k nalezení v adresáři `binaries/atstk1002`. V závislosti na konfiguraci Buildroot zde může být uložen obraz `rootfs.avr32_nofpu.jffs2` souborového systému jffs2 (The Journaling Flash File System version 2), určený do paměti Flash na základní desce STK1000. Jako další tam je obraz `rootfs.avr32_nofpu.ext2` souborového systému ext2 (The Second Extended File System), který je nahráván do paměťové karty SD/MMC. Vedle toho jsou tu i archivy těchto obrazů, archiv `rootfs.avr32_nofpu.tar` s adresářovou strukturou všech souborů, například pro zavedení systému ze sítě. Dalším souborem je `linux-kernel-2.6.23-avr32` s jádrem Linux, které je ovšem už zahrnuto také v předchozích souborech a není proto nutno jej dodatečně kopírovat. Ještě je zde obraz zavaděče U-Boot `u-boot.bin`, pomocí něhož lze nahradit původní uložený v paměti Flash STK1000. Protože jsem chtěl použít právě paměťovou kartu, jako zařízení pro zavedení operačního systému, připojil jsem obraz `rootfs.avr32_nofpu.ext2` k dočasnému adresáři souborového systému hostitele:

```
mkdir /tmp/avr32_image
sudo mount -o loop binaries/atstk1002/rootfs.avr32_nofpu.ext2
/tmp/avr32_image
```

Zapojil jsem čtečku s kartou SD/MMC k USB rozhraní a naformátoval jsem paměťovou kartu pro souborový systém ext2:

```
sudo /sbin/mkfs.ext2 /dev/sda1
```

Naformátovanou kartu jsem připojil k hostitelskému souborovému systému:

```
sudo mount /dev/sda1 /mnt/usbdisk
```

Zkopíroval jsem celou adresářovou strukturu obrazu ext2 do paměťové karty:

```
sudo cp -a /tmp/avr32_image/* /mnt/usbdisk
```

Odpojil jsem paměťovou kartu i obraz ext2 pro cílové zařízení od hostitelského systému:

```
sudo umount /dev/sda1
sudo umount /tmp/avr32_image
```

3.3 Zprovoznění zvukového modulu

Poté co byl vytvořen a nahrán souborový systém ext2 do paměťové karty SD/MMC, jsem mohl konečně začít s oživováním zvukového modulu ve vývojovém kitu. Nejdříve ale bylo potřeba zkontrolovat případně změnit proměnné prostředí zavaděče U-Boot, převážně cestu k obrazu jádra Linux a nastavení správného zaváděcího zařízení:

```
printenv
```

```
bootcmd=
mmcinit; ext2load mmc 0:1 0x90400000 /boot/uImage; bootm 0x90400000

bootargs=
console=/dev/ttyS0 root=/dev/mmcblk0p1 fbmem=600k rootdelay=1
```

Když bylo všechno v pořádku, mohl OS Linux být spuštěn z paměťové karty. Prvním pokusem bylo zavést modul `snd-atmel-ac97`:

```
modprobe snd-atmel-ac97
```

Avšak zde došlo na dlouhou dobu k zablokování systému, protože ovladač během své inicializace zkoušel detekovat zvukový modul, který neodpovídal. Po přidání ladících výpisů do zdrojového kódu `sound/pci/ac97/ac97_codec.c` ovladače a jeho opětovném přeložení jsem zjistil, že se pokoušel ve funkci `snd_ac97_mixer` přečíst výrobní model kodeku a další registry s výchozím nastavením parametrů, ale všechny pokusy o přístup do registrů selhávaly vypršením časového intervalu při čekání na odezvu. Z toho vyplynulo, že zřejmě kodek vůbec s řadičem po rozhraní AC-link nekomunikuje a bylo podezření na hardwarovou chybu na straně kodeku ve zvu-

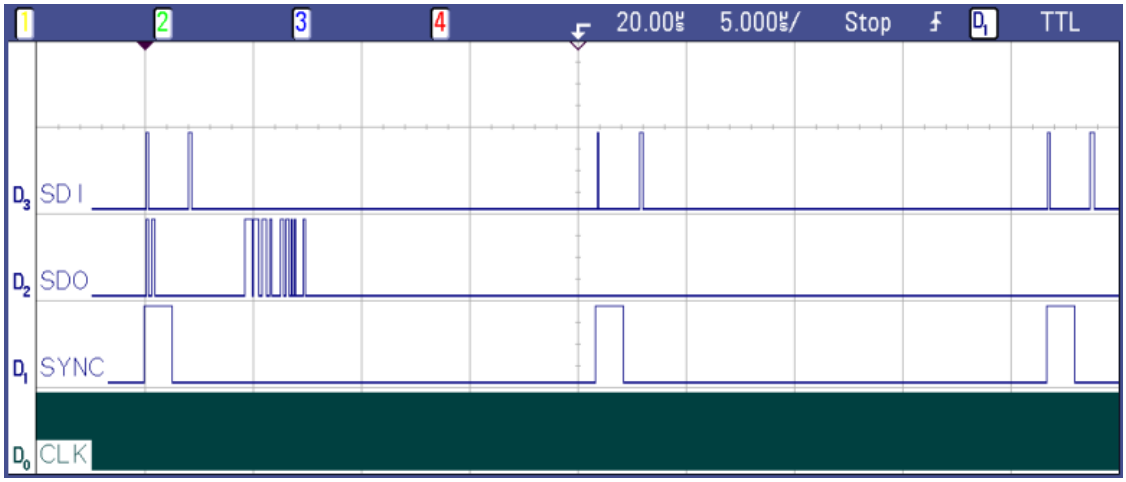
kovém modulu nebo na straně řadiče v jednočipovém systému AP7000, použitém ve vývojovém kitu. Po připojení logického analyzátoru na vodič bitové synchronizace BIT_CLK sběrnice AC-link nebyly k vidění žádné očekávané impulsy. Tím bylo jasné, že problém začíná už ve zvukovém modulu, protože o generování BIT_CLK se stará kodek, pokud tedy není tento signál zkratován. Přiložením sondy osciloskopu přímo na vývod krystalového taktovacího generátoru kodeku bylo jasné, že oscilátor sám o sobě nekmitá. Zvuková karta byla vyjmuta ze základní desky STK1000 a napojena na vnější zdroj stejnosměrného napětí. Na napájecí napětí 3,3 V byl také přes zdvihací rezistor připojen přívod signálu RESET. Je to z toho důvodu, že vstup RESET je obvykle asynchronní a také opačné logiky, t.j. je aktivní ve stavu logické „0“ a za normálního režimu na něm musí být úroveň logické „1“. Tentokrát už se spustilo generování synchronizace BIT_CLK a když se propojily všechny datové vodiče rozhraní AC-link mezi zvukovým modulem a STK1000, byl ovladač `snd-atmel-ac97` schopen při inicializaci detekovat a přidat do seznamu zvukových zařízení naši kartu. Po zasunutí zvukové karty zpátky do rozšiřujícího slotu na základní desce STK1000, přestala opět generovat hodinový kmitočet.

Nakonec se zjistilo, že opravdu byl problém v zapojení vstupu RESET kodeku na vodič se signálem stejného významu ze sběrnice základní desky STK1000. Zřejmě nesmí být aktivní signál RESET hned po zapnutí napájení a v STK1000 stejně jako jinde v mikroprocesorové technice je běžnou praxí generovat tento signál pomocí RC obvodu, kde bezprostředně po zapnutí je na kapacitoru nulové napětí, RESET s opačnou logikou je tedy aktivní a nabíjením kapacitoru se po překročení rozhodovací úrovně deaktivuje. Na plošném spoji testovaného vzorku zvukového modulu jsem proto přerušil vstup signálu RESET z STK1000 a namísto toho jsem jej připojil přes zdvihací rezistor na vodivou cestu s napájecím napětím 3,3 V pro číslicovou část kodeku.

Protože je před každým následujícím zavedením modulu `snd-atmel-ac97` potřeba znovu iniciovat kodek, je také dodatečně na zvukovou kartu přidán mikrospínač ke generování impulsu připojením vstupu RESET na společný (zemní) vodič. Pro případ, že by se příště realizoval podobný zvukový modul, by bylo vhodné připojit vstup kodeku RESET na jeden z vodičů univerzální sběrnice GPIO a potom během incializace ovladače signál RESET generovat programově prostým nastavením logické „0“ a po jistém zpoždění vrácením zpátky na logickou „1“. To samozřejmě bude vyžadovat zásah do zdrojového kódu ovladače, ale jen tak se zajistí spolehlivá inicializace kodeku a schopnost ovladače jej detekovat při každém zavedení a to i během startu OS Linux. V současné verzi není možné ovladač `snd-atmel-ac97` zakompilovat přímo do monolitického jádra Linux, právě proto, že teprve po jisté době po zapnutí napájení je možné generovat RESET pro kodek a potom je také ovladač schopen detekovat zvukový modul. Opětovný pokus načíst ovladač `snd-atmel-ac97`

selže jestliže není znovu vyvolán RESET na kodeku, když už byl jednou zaveden. O tomto, proti všem zvyklostem nestandardním použitím signálu RESET, nebyla v katalogovém listu kodeku AD1886A zmínka, proto musel být zdvihací rezistor a mikrospínač generující RESET přidán až dodatečně a namísto toho byl v původním návrhu tento signál přiváděn ze sběrnice STK1000.

Na obrázku 3.1 jsou vidět průběhy signálů na jednotlivých vodičích rozhraní AC-link v průběhu trvání dvou rámců TDM, tak jak byly zachyceny logickým analyzátorem, když správně probíhala komunikace mezi kodekem na zvukové kartě a řadičem v AP7000 ve vývojovém kitu.

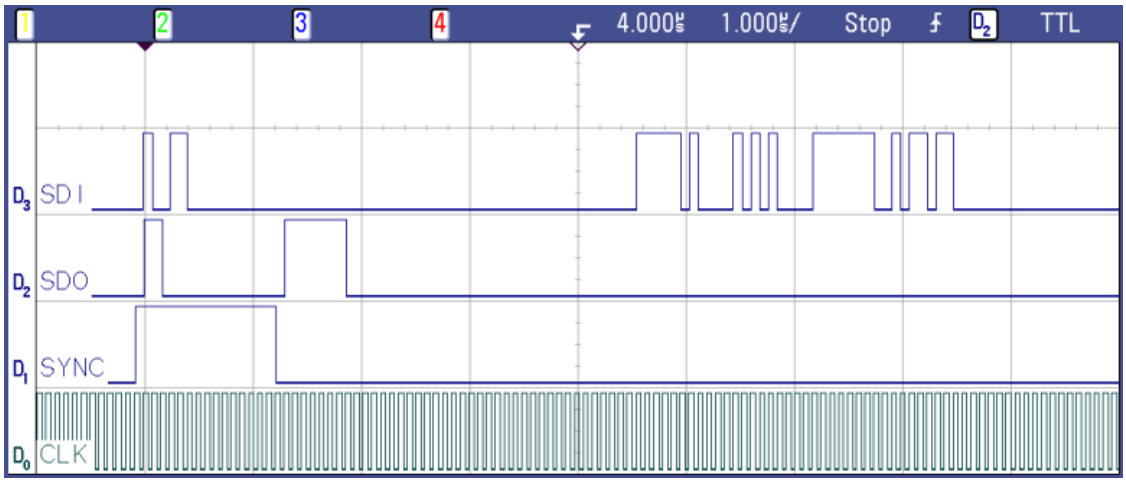


Obr. 3.1: Zachycené průběhy signálů AC-link dvou celých rámců TDM.

Konkrétně je zde zhuštěně zobrazen periodický hodinový signál 12,288 MHz bitové synchronizace CLK generovaný kodekem a rámcová synchronizace SYNC s opakovacím kmitočtem 48 kHz, trvající po dobu příznakové fáze rámce (během časového slotu 0). Výstupní rámcová synchronizace SYNC je odvozena řadičem ze vstupní bitové synchronizace CLK. Na jiném vstupu analyzátoru byly zachyceny shluky pulsů výstupních dat SDO v prvním rámci, reprezentující vzorky PCM levého a pravého kanálu ve slotech 3 a 4. Validita celého rámce SDO je v příznakovém slotu 0 signalizována hned prvním pulsem (bit 15). Platnost dat potom ve slotech 3 a 4 příslušnými bity 12 a 11, tvořícími druhý puls také ve slotu 0 rámce SDO. Směrem od kodeku přichází vstupní data SDI, kde příznakový slot 0 obsahuje jako první puls (bit 15) reprezentující připravenost kodeku a ve slotu 1 širší puls (bity 11 a 10) indikující, že je nastaven režim VRA pro levý a pravý zvukový kanál. To znamená, že číslicově-analogové převodníky pro oba kanály jsou aktivní, ale pracují s jiným vzorkovacím kmitočtem než je standardních 48 kHz a kodek momentálně nevyžaduje po řadiči další vzorky PCM pro tyto převodníky. Druhý rámec SDO, narozdíl od prvního,

neobsahuje vzorky PCM ani jiná data pro kodek, proto chybí i příznak validity celého rámce v prvním bitu nultého slotu. Druhý rámec SDI znovu přenáší příznaky připravenosti kodeku a režim VRA pro levý a pravý kanál, podobně jako v prvním rámci.

Další obrázek 3.2 ukazuje detailněji průběhy pro prvních pět časových slotů. Tady se ovladač `snd-atmel-ac97` pokoušel právě detekovat model kodeku použitém ve zvukovém modulu.



Obr. 3.2: Zachycené průběhy signálů AC-link s pěti sloty.

V příznakovém slotu 0 je tedy impuls vedle bitu 15 validity celého výstupního rámce SDO také rozšířen o pozici bitu 14, signalizujícího platnost příkazového slotu 1. Široký impuls v příkazovém slotu 1 sděluje kodeku, že chce provést operaci čtení (příznakový bit 19) z registru daného indexem, který je určen bity 18–12. Konkrétně na zachyceném průběhu SDO byl vyslán příkaz ke čtení identifikátoru „Vendor ID2“ z registru kodeku s indexem 7Eh. Na tento příkaz musí v příštím rámci SDI přijít odpověď s indexem registru 7Eh ve slotu 1 a kódem výrobního modelu kodeku ve slotu 2. Pro AD1886A by to měl podle katalogového listu [7] být kód 5363h a po dotazu na jiný registr s indexem 7Ch by kodek měl vrátit identifikátor „Vendor ID1“ s kódem 4144h výrobce Analog Devices, Inc. Z obou identifikátorů ovladač `snd-atmel-ac97` přesně určí o jaký kodek se jedná a zvolí vhodnou konfiguraci, šitou na míru tomuto integrovanému obvodu a navíc bude schopen využít všechny vlastnosti kodeku na maximum. Rámec SDI tentokrát obsahuje vzorky PCM levého a pravého kanálu ve slotech 3 a 4.

3.4 Testování s aplikacemi

Nyní nastal čas vyzkoušet zvukový modul také s programy v uživatelském prostoru operačního systému Linux. Bez nastavení parametrů zvuku je implicitně pro všechny výstupy aktivována funkce Mute a není proto nic slyšet. K tomu, aby se nastavily potřebné vlastnosti zvuku je možné využít programu `alsamixer`. Tento nástroj, dodávaný přímo ke zvukovému podsystému ALSA, má uživatelské prostředí řešeno jako virtuální mixážní pult. Nastavení všech parametrů je potom nutno uložit do konfiguračního souboru `/etc/asound.state`, aby bylo nastavení po příštím restartu obnoveno. Dá se tak učinit z příkazového řádku pomocí programu `/usr/sbin/alsactl`:

```
alsactl store
```

Protože se v mém případě modul `snd-atmel-ac97` musí zavádět ručně, potom se k obnovení předchozích uložených nastavení po inicializaci a registraci zvukové karty v podsystému ALSA může zavolat příkaz:

```
alsactl restore
```

Po vytvoření obrazu Linux pro STK1000 jsou implicitně v adresáři `/etc/init.d` skripty volané během startu a ukončení běhu operačního systému. Jsou tam také dva pro automatické obnovení a uložení zvukových nastavení. Protože se v našem případě musí zvuková karta zaregistrovat ručně až bude systém spuštěn, vrací program `alsactl`, automaticky volaný ze skriptu, při startu OS Linux chybu. Také je zbytečné pokoušet se ukládat nastavení zvuku, když při ukončení systému nebyl ani modul použit nebo nebyly nastaveny správné parametry. Proto jsem oba tyto skripty schoval před jejich automatickým voláním, tím že jsem je přejmenoval:

```
mv /etc/init.d/S50alsa-utils /etc/init.d/_S50alsa-utils
mv /etc/init.d/K20alsa-utils /etc/init.d/_K20alsa-utils
```

V následujích řádcích při popisu testování schopností zvukového modulu zaznamenávat a přehrávat zvuková data, budu pro názornost uvádět, jak jsem pracoval se zvukem pomocí programů z příkazového řádku, namísto aplikací s interaktivním uživatelským rozhraním. Nejdříve jsem testoval mikrofonní vstup MIC_IN a nízkoimpedanční výstup do sluchátek HP_OUT. Ze všeho nejdříve jsem musel deaktivovat funkci Mute a dle potřeby také nastavit vhodnou hlasitost do sluchátek (hodnota v procentech):

```
amixer set Headphone 85% unmute
```

Pro nahrávání z mikrofonu musí být přepnut záznam zvuku na mikrofonní vstup:

```
amixer set Mic cap
```

Pokud jsem chtěl také slyšet zvukový signál z mikrofonu v reálném čase přímo přes směšovací obvody kodeku ve zvolené hlasitosti v procentech stačilo vhodně zkombinovat parametry předchozího příkazu:

```
amixer set Mic 80% unmute cap
```

Citlivost mikrofonu zde závisí nejen na nastavených úrovních připoslechu nebo záznamu zvuku, ale mikrofonní vstup je navíc vybaven dvěma předzesilovači, každý s napěťovým zesílením 10 (20 dB). Vnější předzesilovač je tvořen invertujícím operačním zesilovačem IC2 na zvukové kartě. Propojkou JP1 je možno přepínat mezi výstupem z tohoto předzesilovače nebo nezesíleným mikrofonním vstupem z konektoru MIC_IN. Druhý předzesilovač je uvnitř kodeku a lze ho programově aktivovat nebo vyřadit nastavením odpovídajícího bitu v registru o indexu 0Eh. K aktivaci vnitřního mikrofonního předzesilovače jsem použil příkaz:

```
amixer set 'Mic Boost (+20dB)' unmute
```

Za současného použití vnějšího předzesilovače byl mikrofonní vstup velmi citlivý. Zisk signálu z mikrofonu byl teď navýšen 100x (o 40 dB). Pro další testování jsem vnitřní předzesilovač v kodeku vyřadil:

```
amixer set 'Mic Boost (+20dB)' mute
```

Stačilo používat pouze vnější předzesilovač, i tak se později ukázalo, že se musí volit menší úroveň nastavení zisku záznamu, aby digitalizovaný zvuk nebyl zkreslen. Zisk záznamu, tedy úroveň signálů vstupujících do analogově-číslicových převodníků, se nastavuje příkazem:

```
amixer set Capture 50%
```

Po nastavení všech potřebných parametrů jsem mohl vyzkoušet zaznamenat zvuk z mikrofonu do souboru:

```
arecord test.wav
```

Zvuk byl nahrán do souboru implicitně jako monofonní, 8-bitový a se vzorkovacím kmitočtem 8 kHz. Teď jsem mohl ztlumit odposlech mikrofonního vstupu abych slyšel pouze obsah přehrávaného souboru:

```
amixer set Mic mute
```

K tomu, aby byly slyšet výstupy číslicově-analogových převodníků do kterých vstupují vzorky PCM přenesené číslicovým rozhraním AC-link, musí se i tady deaktivovat funkce Mute:

```
amixer set PCM 70% unmute
```

Potom už nic nebránilo tomu poslechnout si co se nahrálo do souboru:

```
aplay test.wav
```

V dalším kroku jsem testoval stereofonní vstup LINE_IN, který jsem připojil k vnějšímu zdroji audio signálu. Přepnul jsem vstupy analogově-číslicových převodníků pro záznam z LINE_IN, včetně připoslechu, příkazem:

```
amixer set Line 40% unmute cap
```

Mohl jsem začít nahrávat:

```
arecord -f S16_LE -c2 -r48000 > test.wav
```

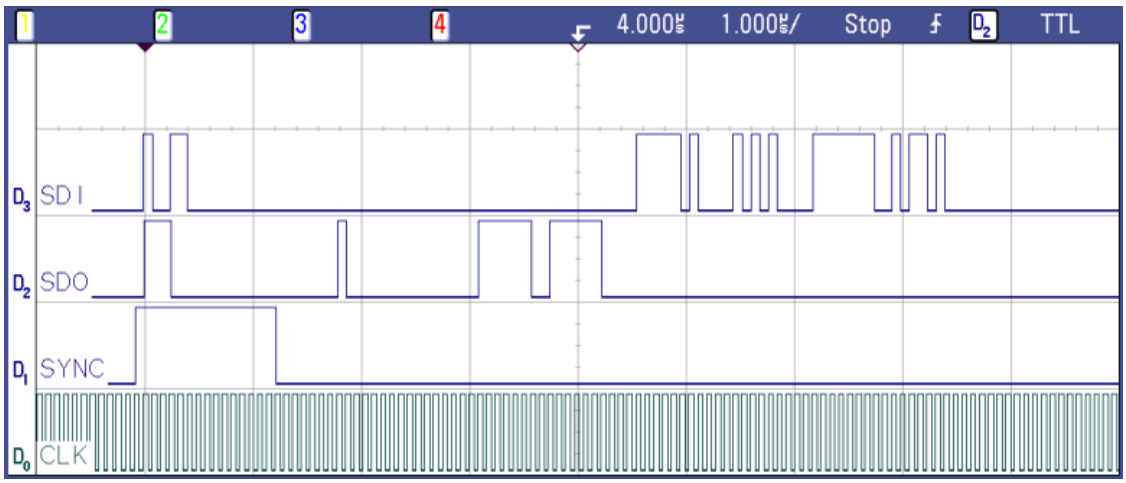
Teď byl zvuk zaznamenán stereofonně, vzorkovacím kmitočtem 48 kHz, ve formátu 16 bitů se znaménkem a s nejméně významným bitem na posledním místě. Opět jsem ztlumil odposlech vstupu a přehrál nový záznam:

```
amixer set Line mute
```

Nakonec přišel na řadu test linkového výstupu LINE_OUT. Výstup jsem zapojil na vnější zesilovač a bylo nutné, jako obvykle, také deaktivovat funkci Mute:

```
amixer set Master 80% unmute
```

Během přehrávání dříve zaznameného souboru byl slyšet výstup podle očekávání. Na následujícím obrázku 3.3 je logickým analyzátorem zachycen průběh komunikace po rozhraní AC-link, když je právě nastavován výstup LINE_OUT.



Obr. 3.3: Zachycené průběhy signálů AC-link při nastavování LINE_OUT.

Je posílán příkaz zápisu parametru „Master Volume“ do registru kodeku. Index 02h registru, jehož obsah se má modifiovat, je přenášen v příkazovém slotu 1 výstupního rámce SDO. Protože se jedná o operaci zápisu, není tentokrát v příkazovém slotu bit 19 nastaven. 16-bitové slovo s novým nastavením je přenášeno ve slotu 2, zbývající 4 nejméně významé pozice slotu jsou nevyužity. V katalogovém listu [7] kodeku AD1886A je stejně jako v dokumentu [6] specifikace AC ‘97 formát slova uloženém v registru 02h takový, že na první pozici (bit 15) je ovládání funkce Mute, bit 14 je nevyužit, bity 13–8 představují úroveň levého kanálu, bity 7 a 6 jsou nevyužity a bity 5–0 nesou hodnotu úrovně pravého kanálu. Impuls na začátku příznakové fáze rámce SDO signalizuje vedle validity celého rámce, také naplněnost slotů 1 a 2 platnými daty. Směrem od kodeku přichází k řadiči vstupní rámec SDI, právě přenášející vzorky PCM levého a pravého kanálu ve slotech 3 a 4.

Zvukový modul jsem také testoval přehráváním souboru formátu MP3 pomocí programu `mplayer`, který je také možné spouštět z příkazové řádky:

```
mplayer smash.mp3
```

Ukázalo se, že kvalita přehrávání souboru MP3 přes zvukový modul se jevila srovnatelná, jako kdyby byla nahrávka přehrávána běžnou zvukovou kartou v osobním počítači.

4 ZÁVĚR

Po seznámení se s technickým i programovým vybavením vývojového kitu STK1000 a hlubším prozkoumání jeho zvukových možností, byla část práce zaměřena na studium standardu AC '97, včetně rozhraní AC-link, které bylo použito ke komunikaci mezi vývojovým kitem a zvukovým modulem.

Potom následoval návrh a realizace zvukového modulu jako zásuvná karta do rozšiřujícího slotu na základní desce STK1000.

Dalším krokem bylo vytvoření spustitelného obrazu souborového systému ext2, který by bylo možné nahrát do paměťové karty SD/MMC, aby potom při startu STK1000 mohl být z této karty zaveden OS Linux s nainstalovaným zvukovým podsystémem ALSA a potřebným ovladačem `snd-atmel-ac97` řadiče číslicového rozhraní AC-link, s podporou použitého kodeku AD1886A. Původní myšlenkou bylo vytvořit obraz standardně podle návodů v uživatelské příručce [1] k dodávanému Linux BSP, přímo od výrobce Atmel. Avšak po potížích s instalací potřebných nástrojů BSP (nefungující odkazy na Internet ke stažení instalačních souborů, nekompatibilita spustitelných souborů a knihoven nezbytných k překladu a sestavení všech komponent obrazu cílového systému), jsem využil automatického generování všech součástí pomocí projektu Buildroot. Nakonec se mě za pomocí Buildroot podařilo potřebný obraz pro paměťovou kartu SD/MMC vytvořit a bylo možné začít se zprovozňováním zvukového modulu.

Zde se ovšem objevil problém, že kodek ve zvukovém modulu negeneroval potřebnou bitovou synchronizaci BIT_CLK a operační systém Linux byl při inicializaci ovladače `snd-atmel-ac97` na delší dobu zablokován, protože při detekci zvukové karty neprobíhala komunikace mezi řadičem AC-link a kodekem. Zjistilo se, že příčinou bylo nestandardní použití signálu RESET v integrovaném obvodu AD1886A, o kterém nebyla v katalogovém listu zmínka a předpokládal jsem, že bude stačit použít RESET ze sběrnice základní desky STK1000, který také pracuje se stejnou logikou, kdy je aktivní ve stavu logické „0“. Signál RESET v STK1000 je aktivní ihned po zapnutí napájení a kodek AD1886A ve zvukovém modulu zřejmě nedokáže tento signál správně zpracovat. Přerušil jsem tedy vstup signálu RESET z STK1000 a namísto toho jsem jej dodatečně připojil přes zdvihací rezistor na napájecí napětí 3,3 V. Také jsem přidal na zvukovou kartu mikrospínač, aby bylo možné kdykoliv při běhu systému aktivovat RESET, krátkodobým připojením na společný vodič (logickou „0“).

Na fotce výsledného zvukového modulu v příloze D.5 je dole vidět dodatečně připojený zdvihací rezistor a mikrospínač ke generování RESET kodeku.

Po správné inicializaci kodeku signálem RESET, ovladač `snd-atmel-ac97` je schopen detekovat zvukovou kartu a zaregistrovat ji v podsystému ALSA. Další testování zvukového modulu zachytáváním audio signálu z mikrofonu nebo linkového vstupu do souboru, jakož i následné přehrávaní záznamu do sluchátek nebo linkového výstupu, potom už probíhalo bez komplikací s očekávaným výsledkem.

Zvukový modul jsem chtěl také otestovat s aplikací VoIP, softwarovým telefonem Linphone, který je možné také spouštět jako konzolovou aplikaci Linux, ale díky výše uvedeným problémům s generováním nástrojů pro překlad a sestavování binárních souborů, stejně jako i kvůli silné závislosti korektního chování Linphone v cílovém systému na jeho konkrétní verzi, už nebyla pro nedostatek času příležitost jej zdárně sestavit pro STK1000.

Cílem této práce bylo navrhnout a zprovoznit zvukový modul pro vývojový kit STK1000 v operačním systému Linux s ovladači ALSA, což se podařilo. Předmětem další práce proto může být například vývoj aplikace pro přenos hlasu po síti TCP/IP, s využitím STK1000 a vytvořeného zvukového modulu.

LITERATURA

[1] Atmel® *AVR32 Linux BSP User Guide* [online]. Version 2.0.0, březen 2007 [cit. 14. dubna 2008]. Dostupné z: <http://www.atmel.com/dyn/resources/prod_documents/avr32_linux_user_guide_2.0.0.tar.gz>.

[2] Atmel® *AT32STK1000 Schematics* [online]. Říjen 2006 [cit. 2. prosince 2007]. Dostupné z: <http://www.atmel.com/dyn/resources/prod_documents/AT32STK1000_schematics.pdf>.

[3] Atmel® *ATSTK1000, Product Card* [online]. ©2008 [cit. 25. května 2008]. Dostupné z: <http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3918>.

[4] Atmel® *AT32AP7000 Preliminary* [online]. Revision K, říjen 2007 [cit. 6. prosince 2007]. Dostupné z: <http://www.atmel.com/dyn/resources/prod_documents/doc32003.pdf>.

[5] Atmel® *AT32AP7000, Product Card* [online]. ©2008 [cit. 19. května 2008]. Dostupné z: <http://www.atmel.com/dyn/products/product_card.asp?part_id=3903>.

[6] Intel® *AC'97 Component Specification* [online]. Revision 2.2, září 2000 [cit. 11. května 2008]. Dostupné z: <http://download.intel.com/support/motherboards/desktop/d201gly/sb/ac97r22.pdf>.

[7] Analog Devices, Inc. *AD1886A AC'97 SoundMAX® Codec, Data Sheet* [online]. Únor 2002 [cit. 13. května 2008]. Dostupné z: <http://www.analog.com/UploadedFiles/Data_Sheets/AD1886A.pdf>.

[8] Atmel® *Atmel AVR32 - open source - Buildroot for AVR32* [online]. ©2007 [cit. 4. května 2008]. Dostupné z: <http://www.atmel.no/buildroot/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AC '97 Audio Codec '97 – specifikace Intel® pro audio kodeky

ADC Analog to Digital Converter – analogově-číslicový převodník

ALSA Advanced Linux Sound Architecture – pokročilý zvukový podsystém operačního systému Linux

API Application Programming Interface – rozhraní k programování aplikací

BSP Board Support Package – podpůrný balíček k hardwaru

DAC Digital to Analog Converter – číslicově-analogový převodník

DSP Digital Signal Processing – číslicové zpracování signálů

GCC GNU Compiler Collection – sbírka překladačů programovacích jazyků projektu GNU

GDB GNU Project Debugger – ladící nástroj projektu GNU

GNU GNU's Not Unix – projekt svobodného softwaru inspirovaný operačním systémem UNIX®

GPIO General Purpose Input Output – vstup/výstup pro všeobecné použití

I^2C Inter-IC – dvoudrátová sběrnice standardu Philips Semiconductors

I^2S Inter-IC Sound – třídrátová sběrnice standardu Philips Semiconductors navržená pro přenos zvukových dat

MMC MultiMediaCard – typ paměťové karty Flash

OSS Open Sound System – zvukový podsystém operačních systémů UNIX/Linux

PCI Peripheral Component Interconnect – standard sběrnice pro připojení periferních zařízení

PCM Pulse Code Modulation – pulsně kódová modulace

RISC Reduced Instruction Set Computer – systém s omezenou instrukční sadou

SD Secure Digital – typ paměťové karty Flash

SDRAM Synchronous Dynamic Random Access Memory – synchronní paměť s náhodným přístupem

SIMD	Single Instruction, Multiple Data – paralení zpracování více toků dat jednou instrukcí
S/PDIF	Sony/Philips Digital Interconnect Format – sériové rozhraní pro přenos zvukových dat
SPI	Serial Peripheral Interface – sériové rozhraní pro periferie
SSC	Synchronous Serial Controller – řadič synchronní sériové komunikace
TDM	Time Division Multiplexing – časový multiplex
TWI	Two-Wire Interface – dvoudrátové rozhraní
USART	Universal Synchronous/Asynchronous Receiver/Transmitter – univerzální synchronní/asynchronní přijímač/vysílač
VoIP	Voice over Internet Protocol – přenos hlasu po sítích se spojováním paketů
VRA	Variable Rate Audio – režim kodeku umožňující změnit vzorkovací kmitočet

SEZNAM PŘÍLOH

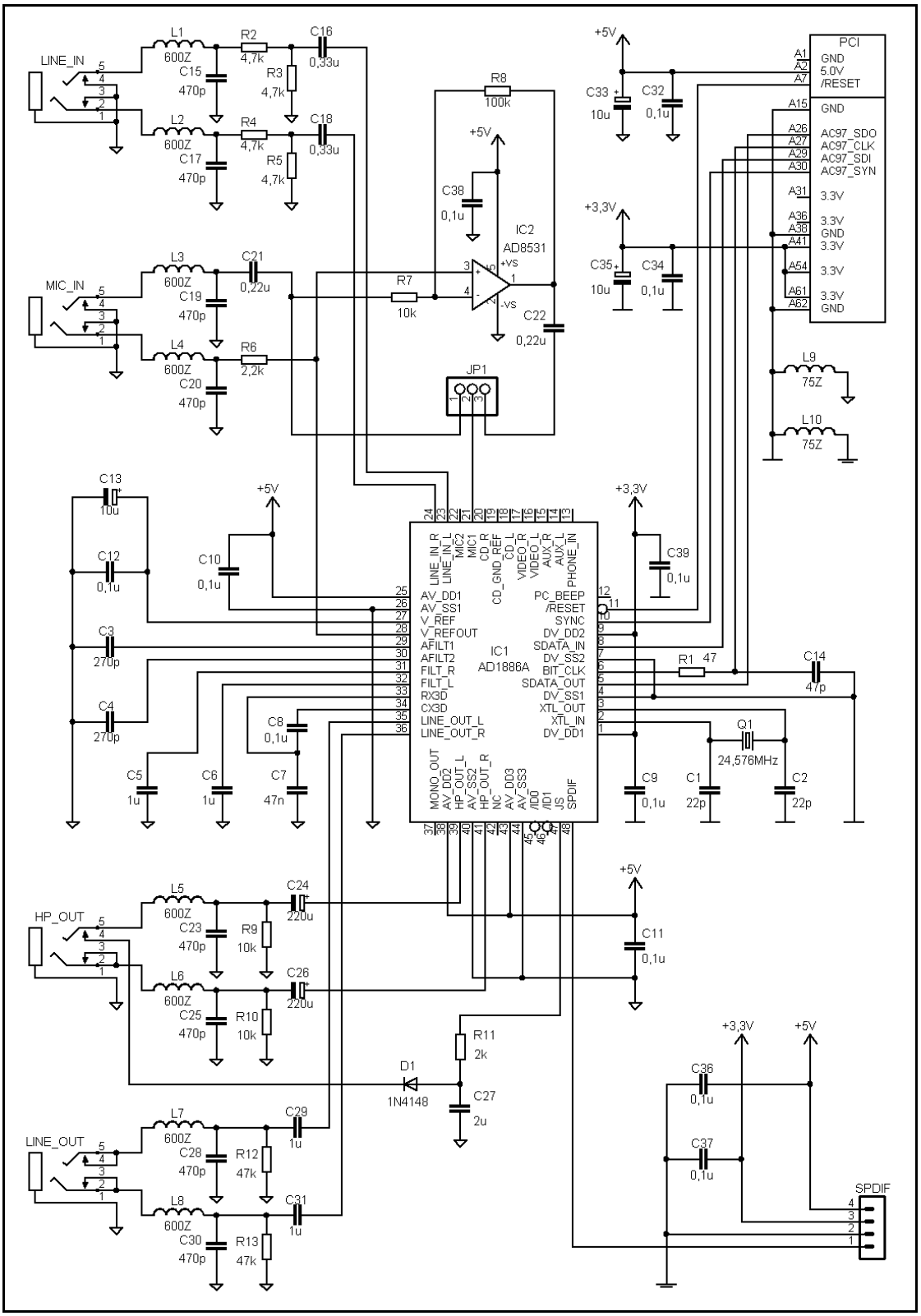
A OBSAH PŘILOŽENÉHO CD

```
/
|-- documents/
|   |-- atmel/
|   |   |-- ap7000/                          dokumentace k AP7000
|   |   `-- stk1000/                         dokumentace k STK1000
|   |-- linux/
|   |   |-- alsa/                            návody ALSA
|   |   |-- sound_howto.pdf                  návod ke zvukové podpoře Linux
|   |   `-- sound_playing_howto.pdf          aplikace pro přehrávání zvuku
|   |-- ac97r22.pdf                          specifikace AC'97, revize 2.2
|   |-- ac97r23.pdf                          specifikace AC'97, revize 2.3
|   |-- ad1886a.pdf                          katalogový list kodeku AD1886A
|   |-- ad8531_32_34.pdf                     katalogový list OZ řady AD853x
|   |-- hdaudio.pdf                          specifikace High Definition Audio
|   `-- murata_emifil.pdf                    katalog tlumivek Murata řady BLM
|-- snd_module/                              soubory Eagle zvukového modulu
|-- software/
|   |-- buildroot/
|   |   |-- configs/
|   |   |   |-- buildroot.config.jp          upravená konfigurace Buildroot
|   |   |   |-- buildroot.config.orig        původní konfigurace Buildroot
|   |   |   |-- linux-2.6.23.config.jp       upravená konfigurace jádra Linux
|   |   |   `-- linux-2.6.23.config.orig     původní konfigurace jádra Linux
|   |   |-- package/                         soubory Makefile programů Linux
|   |   |-- src/
|   |   |   `-- dl/                          lokální depozitář součástí projektu
|   |   `-- buildroot-avr32-v2.1.0.tar.bz2   instalace Buildroot
|   |-- eagle.4.16r2/                        instalace použité verze Eagle
|   `-- images/
|       |-- bsp/                             originální obrazy Linux BSP
|       |-- linux-kernel-2.6.23-avr32        samotné jádro Linux
|       |-- rootfs.avr32_nofpu.ext2          obraz ext2 OS Linux s podporou ALSA
|       |-- rootfs.avr32_nofpu.ext2.bz2      komprimovaný obraz ext2
|       |-- rootfs.avr32_nofpu.tar           archiv všech souborů cíle
|       |-- rootfs.avr32_nofpu.tar.bz2       komprimovaný archiv všech souborů
|       `-- u-boot.bin                       obraz zavaděče U-Boot
|-- metadata.pdf                             popisný soubor bakalářské práce
|-- readme.txt                               soubor s tímto seznamem
`-- thesis.pdf                               text bakalářské práce
```

B ZAPOJENÍ ROZŠIRUJÍCÍHO SLOTU

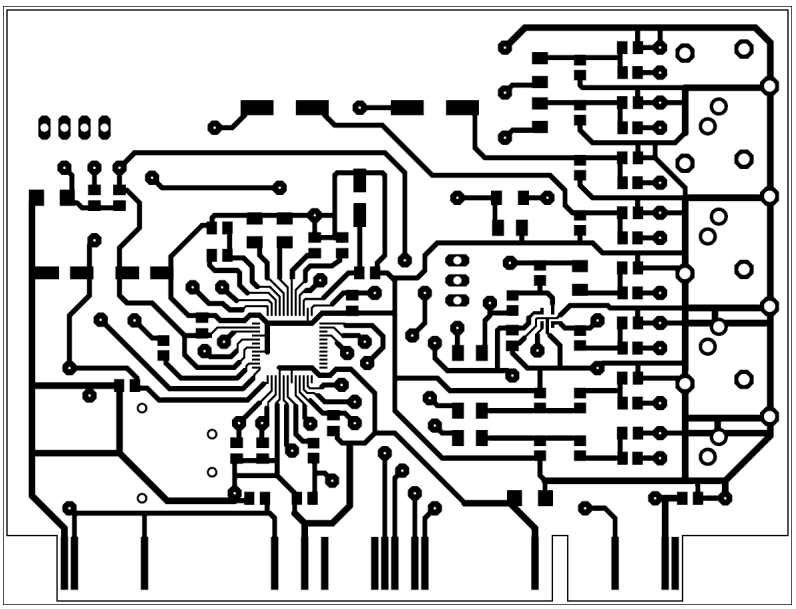
Signál	Pin		Signál	Signál	Pin		Signál
GND	B1	A1	GND	LCD_D20	B49	A49	EBI_A3
3.3V	B2	A2	5.0V	DMARQ_B	B50	A50	EBI_A25
PWM_A	B3	A3	2.5V	DMARQ_C	B51	A51	EBI_NWR3
LCD_CC	B4	A4	EBI_CFCE1	LCD_D21	B52	A52	EBI_A4
PWM_B	B5	A5	EBI_CFCE2	DMARQ_D	B53	A53	EBI_A5
LCD_HSYNC	B6	A6	EBI_A22	LCD_D22	B54	A54	3.3V
LCD_PCLK	B7	A7	RESET_N	LCD_D23	B55	A55	EBI_NCS0
PWM_C	B8	A8	EBI_D0	GND	B56	A56	EBI_NRD
LCD_VSYNC	B9	A9	EBI_D1	TWI_DATA	B57	A57	EBI_NWR1
3.3V	B10	A10	EBI_D2	TWI_CLK	B58	A58	EBI_NCS1
LCD_DVAL	B11	A11	EBI_D3	EIM_NMI_N	B59	A59	EBI_NWAIT
				2.5V	B60	A60	EBI_NWR0
LCD_MODE	B14	A14	EBI_D4	5.0V	B61	A61	3.3V
LCD_PWR	B15	A15	GND	GND	B62	A62	GND
PM_GCLK_B	B16	A16	EBI_D5				
LCD_D0	B17	A17	EBI_D8	ISI_DATA0	B63	A63	EBI_CFRNW
PM_GCLK_C	B18	A18	EBI_D6	ISI_DATA1	B64	A64	EBI_A6
LCD_D1	B19	A19	EBI_D9	ISI_DATA2	B65	A65	EBI_A7
LCD_D2	B20	A20	EBI_D7	ISI_DATA3	B66	A66	EBI_A8
3.3V	B21	A21	PM_GCLK_A	GND	B67	A67	EBI_A9
LCD_D3	B22	A22	EBI_D10	ISI_DATA4	B68	A68	EBI_A10
LCD_D4	B23	A23	TIMER_A_TIOA0	ISI_DATA5	B69	A69	EBI_A11
GND	B24	A24	TIMER_A_TCLK0	ISI_DATA6	B70	A70	EBI_A12
LCD_D5	B25	A25	EBI_D11	ISI_DATA7	B71	A71	EBI_A13
LCD_D6	B26	A26	AC97_SDO	ISI_DATA8	B72	A72	EBI_A14
3.3V	B27	A27	AC97_SCLK	ISI_DATA9	B73	A73	EBI_A15
LCD_D7	B28	A28	EBI_D12	ISI_DATA10	B74	A74	HSR
LCD_D8	B29	A29	AC97_SDI	ISI_DATA11	B75	A75	LCD_VGL
GND	B30	A30	AC97_SYNC	ISI_HSYNC	B76	A76	LCD_5V
LCD_D9	B31	A31	3.3V	ISI_VSYNC	B77	A77	LCD_VGH
LCD_D10	B32	A32	EIM_EXTINT_A	ISI_PCLK	B78	A78	LCD_BACKLIGHT
PM_GCLK_D	B33	A33	SPI_MOSI	GND	B79	A79	LCD_M
LCD_D11	B34	A34	EBI_D13	ISI_MCLK	B80	A80	LCD_VCOM
EIM_EXTINT_B	B35	A35	SPI_MISO	SPI_NPCS0	B81	A81	DAC0
LCD_D12	B36	A36	3.3V	SPI_NPCS1	B82	A82	DAC1
EIM_EXTINT_C	B37	A37	SPI_SCK	SPI_NPCS2	B83	A83	EBI_SDCK
LCD_D13	B38	A38	GND	RTC_WDTEXT	B84	A84	EBI_SDCKE
UART_A_CLK	B39	A39	UART_A_RXD	TIMER_A_TIOB0	B85	A85	HSL
LCD_D14	B40	A40	UART_A_TXD	EIM_EXTINT_E	B86	A86	EBI_NCS2
LCD_D15	B41	A41	3.3V	EIM_EXTINT_F	B87	A87	EBI_NCS3
PM_WAKE_N	B42	A42	EBI_NCS4	EBI_SDA10	B88	A88	EBI_NANDOE
LCD_D16	B43	A43	EBI_D14	EBI_SDWE	B89	A89	EBI_NANDWE
LCD_D17	B44	A44	EBI_NCS5	EBI_CAS	B90	A90	EBI_BA0
DMARQ_A	B45	A45	EBI_A0	EBI_RAS	B91	A91	EBI_BA1
LCD_D18	B46	A46	EBI_D15	1.8V_REF	B92	A92	EBI_SDCS1
LCD_D19	B47	A47	EBI_A1	3.3V	B93	A93	5.0V
GND	B48	A48	EBI_A2	GND	B94	A94	GND

C SCHÉMA ZVUKOVÉHO MODULU

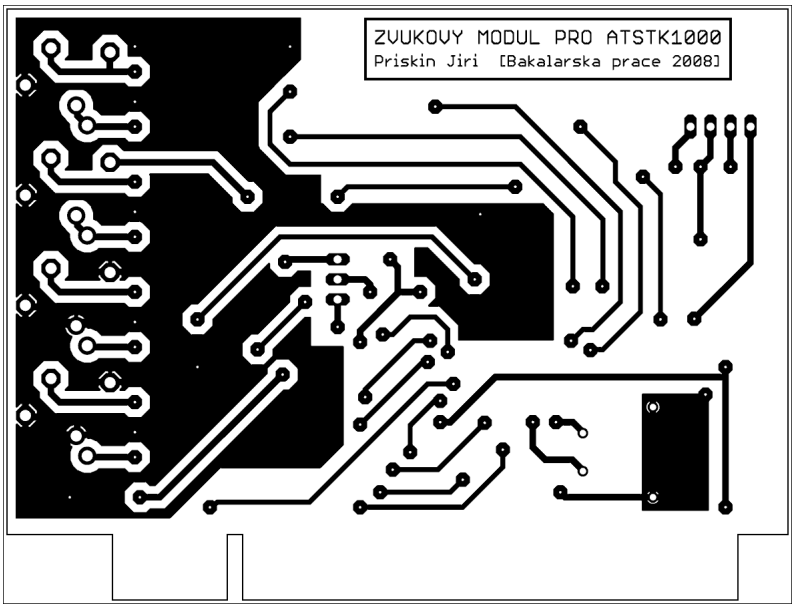


D REALIZACE ZVUKOVÉHO MODULU

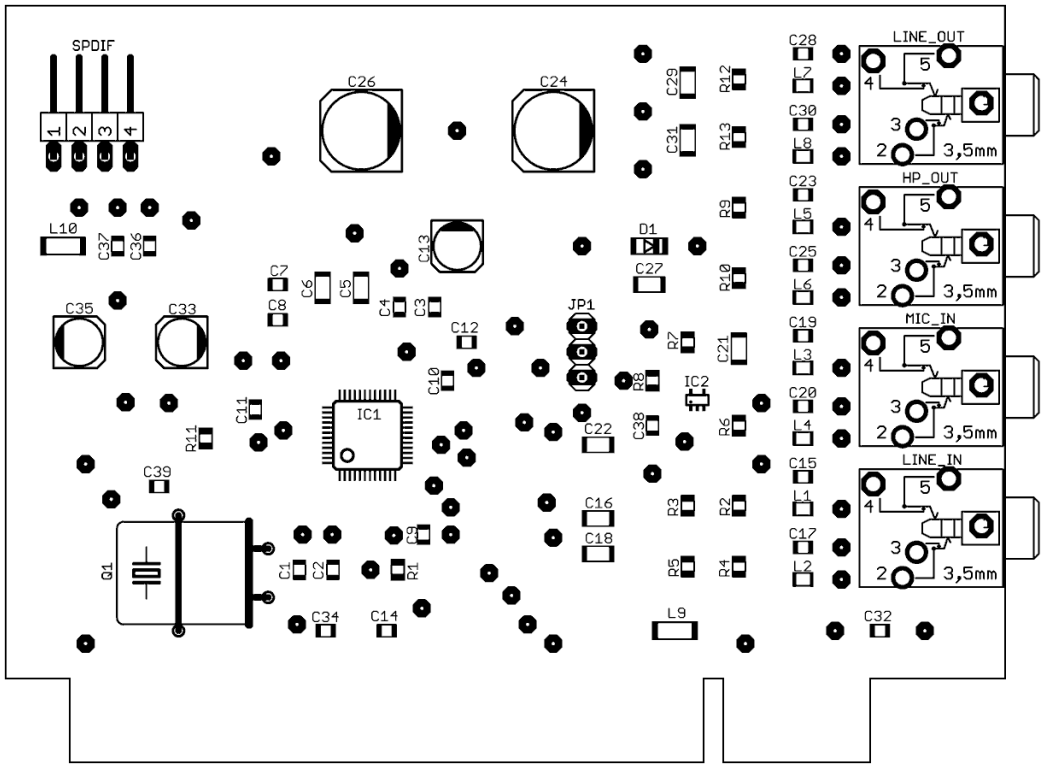
D.1 Motiv plošného spoje ze strany součástek



D.2 Motiv plošného spoje ze strany spojů



D.3 Osazení plošného spoje



D.4 Seznam součástek

R1	47 Ω	SMD 0805
R2	4,7 kΩ	SMD 0805
R3	4,7 kΩ	SMD 0805
R4	4,7 kΩ	SMD 0805
R5	4,7 kΩ	SMD 0805
R6	2,2 kΩ	SMD 0805
R7	10 kΩ	SMD 0805
R8	100 kΩ	SMD 0805
R9	10 kΩ	SMD 0805
R10	10 kΩ	SMD 0805
R11	2 kΩ	SMD 0805
R12	47 kΩ	SMD 0805
R13	47 kΩ	SMD 0805
C1	22 pF	SMD 0805
C2	22 pF	SMD 0805
C3	270 pF	SMD 0805
C4	270 pF	SMD 0805
C5	1 μF	SMD 1206
C6	1 μF	SMD 1206
C7	47 nF	SMD 0805
C8	100 nF	SMD 0805
C9	100 nF	SMD 0805
C10	100 nF	SMD 0805
C11	100 nF	SMD 0805
C12	100 nF	SMD 0805
C13	10 μF/50 V	SMD B
C14	47 pF	SMD 0805
C15	470 pF	SMD 0805
C16	330 nF	SMD 1206
C17	470 pF	SMD 0805
C18	330 nF	SMD 1206
C19	470 pF	SMD 0805
C20	470 pF	SMD 0805
C21	220 nF	SMD 1206
C22	220 nF	SMD 1206
C23	470 pF	SMD 0805
C24	220 μF/16 V	SMD D
C25	470 pF	SMD 0805
C26	220 μF/16 V	SMD D
C27	2 μF	SMD 1206

C28	470 pF	SMD 0805
C29	1 μF	SMD 1206
C30	470 pF	SMD 0805
C31	1 μF	SMD 1206
C32	100 nF	SMD 0805
C33	10 μF/50 V	SMD B
C34	100 nF	SMD 0805
C35	10 μF/50 V	SMD B
C36	100 nF	SMD 0805
C37	100 nF	SMD 0805
C38	100 nF	SMD 0805
C39	100 nF	SMD 0805
L1	600 Ω/100 MHz	SMD 0805 (Murata: BLM 21 . . .)
L2	600 Ω/100 MHz	SMD 0805 (Murata: BLM 21 . . .)
L3	600 Ω/100 MHz	SMD 0805 (Murata: BLM 21 . . .)
L4	600 Ω/100 MHz	SMD 0805 (Murata: BLM 21 . . .)
L5	600 Ω/100 MHz	SMD 0805 (Murata: BLM 21 . . .)
L6	600 Ω/100 MHz	SMD 0805 (Murata: BLM 21 . . .)
L7	600 Ω/100 MHz	SMD 0805 (Murata: BLM 21 . . .)
L8	600 Ω/100 MHz	SMD 0805 (Murata: BLM 21 . . .)
L9	75 Ω/100 MHz	SMD 1806 (Murata: BLM 41 . . .)
L10	75 Ω/100 MHz	SMD 1806 (Murata: BLM 41 . . .)
Q1	24,576 MHz	HC49/U
D1	1N4148	SMD SOD80
IC1	AD1886A	SMD LQFP-48
IC2	AD8531	SMD SC70-5L
LINE_IN	Stereo Jack 3,5 mm, 2 vypínací kontakty	PG203J
MIC_IN	Stereo Jack 3,5 mm, 2 vypínací kontakty	PG203J
HP_OUT	Stereo Jack 3,5 mm, 2 vypínací kontakty	PG203J
LINE_OUT	Stereo Jack 3,5 mm, 2 vypínací kontakty	PG203J
JP1	Lišta jednořadá přímá 2,54 mm, 3 kontakty	1X03
SPDIF	Vidlice zahnutá 2,54 mm, 4 kontakty	MA04-1W

D.5 Zhotovený zvukový modul